AFRL-RY-WP-TR-2017-0074

FULLY ADAPTIVE RADAR MODELING AND SIMULATION DEVELOPMENT

Kristine L. Bell and Anthony Kellems
Metron, Inc.

Graeme E. Smith
The Ohio State University

Bruce L. McKinley
Signal Processing Consultants, Inc.

APRIL 2017
Final Report

THIS IS A SMALL BUSINESS INNOVATION RESEARCH (SBIR) PHASE I REPORT.

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

// Signature//

Muralidhar RangaswamyProgram Manager
Program Manager
Radio Frequency Exploitation Branch

// Signature//

Gregory Cazzell, Chief
Radio Frequency Exploitation Branch

// Signature//

Doug Hager, Deputy
Layered Sensing Exploitation Division

NOTICE TO ACCOMPANY THE DISSEMINATION OF EXPORT CONTROLLED TECHNICAL DATA

Export of the attached information (which includes, in some circumstances, release to foreign nationals within the United States) without first obtaining approval or license from the Department of State for items controlled by the International Traffic in Arms Regulation (ITAR), or the Department of Commerce for items controlled by the Export Administration Regulation (EAR), may constitute a violation of law.

Under 22 U.S.C. 2778, the penalty for unlawful export of items or information controlled under the ITAR is up to ten years imprisonment, or a fine of $1,000,000, or both. Under 50 U.S.C., appendix 2410, the penalty for unlawful export of items or information controlled under the EAR is a fine of up to $1,000,000, or five times the value of the exports, whichever is greater; or for an individual, imprisonment of up to 10 years, or fine of up to $250,000, or both.

In accordance with your certification that establishes you as a "qualified U.S. contractor," unauthorized dissemination of this information is prohibited and may result in your disqualification as a qualified U.S. contractor, and may be considered in determining your eligibility for future contracts with the Department of Defense.

The U.S. Government assumes no liability for direct patent infringement, or contributory patent infringement, or misuse of technical data.

The U.S. Government does not warrant the adequacy, accuracy, currency, or completeness of the technical data.

The U.S. Government assumes no liability for loss, damage, or injury resulting from manufacture or use for any purpose of any product, article, system, or material involving reliance upon any or all technical data furnished in response to the request for technical data.

If the technical data furnished by the Government will be used for commercial manufacturing or other profit potential, a license for such use may be necessary. Any payments in support of the request for data do not include or involve any license rights.

A copy of this notice shall be provided with any partial or complete reproduction of these data that are provided to qualified U.S. contractors.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE ***(DD-MM-YY)*** April 2017	**2. REPORT TYPE** Final	**3. DATES COVERED** ***(From - To)*** 9 June 2016 – 9 January 2017

4. TITLE AND SUBTITLE FULLY ADAPTIVE RADAR MODELING AND SIMULATION DEVELOPMENT	**5a. CONTRACT NUMBER** FA8650-16-M-1774
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 65502F
6. AUTHOR(S) Kristine L. Bell and Anthony Kellems (Metron, Inc.) Graeme E. Smith (The Ohio State University) Bruce L. McKinley (Signal Processing Consultants, Inc.)	**5d. PROJECT NUMBER** 3005
	5e. TASK NUMBER N/A
	5f. WORK UNIT NUMBER Y1HV

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Metron, Inc. 1818 Library St., Suite 600 Reston, VA 20190	The Ohio State University Signal Processing Consultants, Inc.	**8. PERFORMING ORGANIZATION REPORT NUMBER**

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory Sensors Directorate Wright-Patterson Air Force Base, OH 45433-7320 Air Force Materiel Command United States Air Force	**10. SPONSORING/MONITORING AGENCY ACRONYM(S)** AFRL/RYAP
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S) AFRL-RY-WP-TR-2017-0074

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
This is a Small Business Innovation Research (SBIR) Phase I Report. PAO case number 88ABW-2017-1028, Clearance Date 17 March 2017. SBIR data rights waived by contractor. Letter on file. Report contains color.

14. ABSTRACT
This is a Small Business Innovation Research (SBIR) Phase I Report developed under SBIR contract for topic AF161-132 Fully Adaptive Radar Modeling and Simulation Development. We have developed a MATLAB-based modeling and simulation (M&S) architecture for distributed fully adaptive radar (FAR) that will enable algorithm development and testing on simulated, previously collected, and real-time streaming data. The architecture is coded in MATLAB using an object oriented programming approach. The architecture includes a FAR engine to control the operation of the perception-action cycle and software objects that determine the next set of sensing parameters; obtain data from the sensor; process the data to track the target; and store and display the results of the sensing and tracking processes. We have developed modules that implement simulated and pre-recorded software defined radar (SDR) data examples, and real-time and simulated data examples from the Cognitive Radar Engineering Workspace (CREW) at The Ohio State University. The FAR M&S architecture allows for transparent switching between the simulated and experimental CREW data sources, as well as between FAR algorithms that drive the sensing. The ability to easily interchange sensing and processing modules will allow for rapid development and testing of cognitive radar algorithms by structuring the M&S functions to avoid duplicating effort and "single point" solutions. It will enable collaboration between researchers in industry, academia, and the Air Force, as algorithms developed by different researchers can be tested and compared using consistent simulations, collected data, and laboratory conditions. Report contains color.

15. SUBJECT TERMS
SBIR report, fully adaptive radar, cognitive radar, modeling and simulation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT:**	**8. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** (Monitor)
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	50	Muralidhar Rangaswamy **19b. TELEPHONE NUMBER** *(Include Area Code)* N/A

METRON
SCIENTIFIC SOLUTIONS

Metron, Inc
1818 Library Street
Suite 600
Reston, VA 20190
(703) 787-8700
(703) 787-3518 FAX
www.metsci.com

9 February 2017

Dr. Muralidhar Rangaswamy
AFRL/RYAP
2241 Avionics Circle
Wright-Patterson AFB, OH 45433

Subject: Contract Number FA8650-16-M-1774, Phase I SBIR Topic AF161-132
Fully Adaptive Radar Modeling and Simulation Development
Report # TBD

Dear Dr. Rangaswamy:

Metron, Inc. hereby waives its SBIR Data Rights to all contents of the final report for subject contract. The Government is granted an unlimited nonexclusive license to use, modify, reproduce, release, perform, and display or disclose this report and the data contained herein.

Sincerely,

Kristine L Bell

Kristine L. Bell, Ph.D.
Senior Scientist
Advanced Mathematics Applications Division
Metron, Inc.
bell@metsci.com
703-326-2913

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

1 SUMMARY

The potential of cognitive approaches to enhance existing radar performance in almost all respects has led to an upsurge in research in recent years and a key gap in the Air Force's radar modeling and simulation (M&S) tools is the lack of a comprehensive, dynamic distributed radar scenario generation capability for distributed fully adaptive radar (FAR) systems. As of early 2015, the research had all been advancing concepts theoretically and examining their performance through simulation, or at best using pre-recorded data. There had been no reports of experimentally validated concepts, largely because the necessary hardware to test them had not been developed. However, this step is vital in order to establish the true performance potential of applying cognitive processing methods. To address this, the Cognitive Sensing Laboratory (CSL) at The Ohio State University (OSU) ElectroScience Laboratory (ESL), in conjunction with Metron, Inc., the Air Force Research Laboratory (AFRL), and the Air Force Office of Scientific Research (AFOSR), has embarked on a program of research to develop and examine cognitive radar processing concepts analytically and experimentally.

The CSL designed and built the Cognitive Radar Engineering Workspace (CREW), the world's first radar testbed built specifically to allow testing of fully adaptive and cognitive algorithms, and Metron and OSU have developed a theoretical framework for a cognitive FAR system that identifies and mathematically models the key system components within the context of target detection and tracking for a single sensor and target. We have been developing modeling, simulation, analysis, and experimentation capabilities to demonstrate the performance improvement achieved by FAR systems over traditional feed-forward radar (FFR) systems. We began with simulated scenarios and pre-recorded data from OSU's Software Defined Radar (SDR) system. We now have the capability to demonstrate the real-time operation of the cognitive radar tracking system using the CREW.

The goal of this project is to develop a MATLAB-based M&S architecture for distributed FAR radar that will enable algorithm development and testing on simulated, previously collected, and real-time streaming data. In Phase I, we have developed a baseline FAR M&S architecture that is coded in MATLAB using an object oriented programming (OOP) approach. It includes a FAR engine to control the operation of the perception-action (PA) cycle and software objects that determine the next set of sensing parameters; obtain data from the sensor; process the data to track the target; and store and display the results of the sensing and tracking processes. We have developed modules that implement simulated and pre-recorded SDR data examples, and real-time and simulated CREW data examples.

The FAR M&S architecture developed in Phase I allows for transparent switching between the simulated and experimental CREW data sources, as well as between FAR algorithms that drive the sensing. The ability to easily interchange sensing and processing objects will allow for rapid development and testing of cognitive radar algorithms by structuring the M&S functions to avoid duplicating effort and "single point" solutions. It will enable collaboration between researchers in industry, academia, and the Air Force, as algorithms developed by different researchers can be tested and compared using consistent simulations, collected data, and laboratory conditions.

2 INTRODUCTION

Radar systems are crucial for robust surveillance, target acquisition, and reconnaissance in all weather conditions and over wide ranges of interest. Anti-access/area denial (A2/AD) environments are especially challenging, therefore it is necessary to develop innovative signal and data processing techniques to provide advanced sensing capabilities to the warfighter. Distributed multiple input multiple output (MIMO) FAR systems provide the potential to exploit all available degrees of freedom on transmit and receive in order to maximize radar system performance, thus they offer much promise for improved sensing as well as the creation of new sensing modalities.

MIMO radar systems employ multiple transmit and receive elements with transmit elements that have the ability to transmit arbitrary waveforms simultaneously and receive elements that have the ability to process all of the transmitted signals jointly [1]-[10]. Of primary interest are MIMO ground moving target indicator (GMTI) systems consisting of distributed airborne platforms. As in traditional single input single output (SISO) and single input multiple output (SIMO) systems, effective clutter suppression is a key factor in system performance, and the availability of realistic clutter models and simulated clutter data is critical for advanced system design and performance analysis.

M&S tools developed by the Air Force include the Research Laboratory Space Time Adaptive Processing (RLSTAP) Algorithm Development Tool [11], which provides a flexible simulation and analysis tool for SIMO GMTI systems that employ monostatic space-time adaptive processing (STAP) [12]-[14], and the Signal Modeling and Simulation Tool for Multichannel Bistatic Systems (SMS-MBS) [15]-[17], which provides an advanced M&S capability for bistatic SIMO STAP systems. The SMS-MBS tool can generate multiple data cubes for multiple coherent processing intervals (CPIs) with automatically updated bistatic scenarios, but is limited to geometries for which the bistatic angle is small and the monostatic-bistatic equivalence theorem (MBET) approximation is valid. The MIMO Radar Clutter Modeling and Simulation (MIMO-CMS) tool [18],[19], developed by Metron, OSU, and Signal Processing Consultants, Inc. (SPC), provides a physics-based M&S capability for distributed airborne MIMO GMTI systems and extends the M&S capabilities to multistatic STAP [20] and multistatic MIMO [21] configurations. MIMO-CMS accurately characterizes the statistical and spectral properties of the clutter for a variety of radar operating parameters and site-specific geometry and scattering environments. It uses a physics-based bistatic scattering model developed by OSU and is not limited by the MBET approximation [22]. A companion set of toolbox functions calculates fundamental quantities such as iso-range contours, the radar range equation, mean clutter radar cross section (RCS), clutter amplitude and spectral characteristics, the true covariance matrix, and the clutter rank [23]. Additional functions perform MIMO signal processing, calculate performance metrics such as signal-to-noise-ratio (SNR), mean-square error (MSE), detection and false alarm probabilities; and perform statistical and experimental model goodness-of-fit tests. MIMO-CMS currently only provides a single CPI data cube for a fixed set of radar system parameters.

FAR systems, also called cognitive radar (CR) systems, mimic the PA cycle of cognition [24],[25] to adapt the radar sensor in real-time to collect data to achieve a required level of performance [26]-[43]. This requires developing a perception of the current system status, prediction of the effect different sensing actions, and choosing the next sensing action, all in real-time. Haykin introduced the concept of cognitive radar in [26], however the motivation and many of the underlying ideas grew out of the fields of knowledge-aided radar signal processing [41]-[47], agile waveform design [48]-[55], sensor management [56]-[66], sonar ping control [67]-[69], bio-mimetic signal processing [70]-[75], and echoic flow [76],[77]. Although the concept of sensor adaptation goes back several decades, research into truly cognitive systems is currently in its infancy.

The potential of cognitive approaches to enhance existing radar performance in almost all respects has led to an upsurge in research in recent years. As of early 2015, the research had all been advancing concepts theoretically and examining their performance through simulation, or at best using pre-recorded data. There had been no reports of experimentally validated concepts, largely because the necessary hardware to test them had not been developed. However, this step is vital in order to establish the true performance potential of applying cognitive processing methods. To address this, the CSL at the OSU ESL, in conjunction with Metron, Inc., AFRL, and AFOSR, has embarked on a program of research to develop and examine cognitive radar processing concepts analytically and experimentally.

Under an AFOSR-sponsored grant from the 2013 Defense University Research Instrumentation Program (DURIP), the CSL designed and built the CREW, the world’s first radar testbed built specifically to allow testing of fully adaptive and cognitive algorithms. The CREW became operational in early 2015. Under AFRL-sponsored Small Business Innovative Research (SBIR) Topic AF 131-135 Fully Adaptive Radar, Metron and OSU have developed a theoretical framework for a cognitive FAR system that identifies and mathematically models the key system components within the context of target detection and tracking for a single sensor and target [30]-[34]. In the on-going FAR Phase II, we have been investigating theoretical extensions to the FAR framework and developing modeling, simulation, analysis, and experimentation capabilities to demonstrate the performance improvement achieved by FAR systems over traditional FFR systems [35]-[40]. We began with simulated scenarios [31]-[33] and pre-recorded data [34]-[35] from OSU’s SDR system [78]-[84]. Pre-recorded data offers limited opportunities to test cognitive algorithms since the sensing does not truly adapt in real-time, and cognitive processing must be done “after-the-fact” in an artificial manner. We now have the capability to demonstrate the real-time operation of the cognitive radar tracking system using the CREW [36]-[40]. Demonstrations were given to AFRL senior management in October 2015 and May 2016.

A key gap in the Air Force’s previously developed radar M&S tools is the lack of a comprehensive, dynamic distributed radar scenario generation capability for distributed FAR systems. Therefore, the goal of this project (SBIR Topic 161-132 FAR M&S Development) is to develop a MATLAB-based M&S architecture for distributed FAR radar that will enable algorithm development and testing on simulated, previously collected, and real-time streaming data. In Phase I, we have developed a baseline FAR M&S architecture that implements the FAR framework for a stationary sensor system. Throughout this report we use the term “FAR framework” to denote the mathematical model of a FAR system and the term “FAR M&S architecture” to denote the MATLAB-based software implementation of the FAR framework.

The FAR M&S architecture is coded in MATLAB using an OOP approach. It includes a FAR engine to control the operation of the PA cycle and software objects that determine the next set of sensing parameters; obtain data from the sensor; process the data to track the target; and store and display the results of the sensing and tracking processes. We have developed modules that implement simulated and pre-recorded data examples in [30]-[35], and the real-time CREW data examples in [36]-[39]. We have developed a simulation of the CREW and the application programming interface (API) layers for the simulated and experimental CREW data sources to enable switching between simulated and experimental data. A demonstration was given in March 2016 for members of the North Atlantic Treaty Organization (NATO) Sensors Electronics Technology (SET)-227 Panel on Cognitive Radar.

The FAR M&S architecture developed in Phase I allows for transparent switching between the simulated and experimental data sources, as well as between FAR algorithms that drive the sensing. The ability to easily interchange sensing and processing objects will allow for rapid development and testing of cognitive radar algorithms by structuring the M&S functions to avoid duplicating effort and "single point" solutions. It will enable collaboration between researchers in industry, academia, and the Air Force, as algorithms developed by different researchers can be tested and compared using consistent simulations, collected data, and laboratory conditions. In Phase II, we plan to make the FAR M&S architecture code available to members of the NATO SET-227 Panel on Cognitive Radar. Collaborations with members of this panel to develop and test algorithms on the CREW are already underway. Furthermore, several members have already begun development of their own cognitive radar test beds [85],[86] and our FAR M&S architecture will enable further collaboration within the panel using these data sources.

In Phase II, we also plan to extend the baseline architecture to model a dynamic, distributed airborne MIMO radar FAR system using the full MIMO-CMS tool as the simulation base. This will provide a comprehensive radar scenario generation capability that will fill a key gap in the Air Force's previously developed radar M&S tools.

This report is organized as follows. In Chapter 3, we provide an overview of the FAR framework and the applications developed in [30]-[40]. In Section 3.1, we develop the theoretical framework and in Section 3.2, we give an overview of the applications. In Chapter 4, we provide a description of the FAR M&S architecture developed for this project. In Section 4.1 we describe the software structure and modules, in Section 4.2 we provide a guide to using the MATLAB code, in Section 4.3 we give a more detailed description of the base classes, and in Section 4.4, we provide a description of the CREW sensor object. Chapter 5 contains a summary and conclusions.

3 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 Fully Adaptive Radar Framework

3.1.1 General Framework

The heart of cognition is the *perception-action cycle* that both informs and is informed by *memory* [24]-[25]. Cognition requires stimulation by *sensors*. In the human this is via hearing, touch, smell, vision, and taste. The nervous system *processes* the sensed stimuli and converts them into a *perceptio*n of the world. We are able to take informed *action* by interpreting our perception of the world and making decisions. As a consequence, the nervous system sends signals that activate our muscles, thus enabling the desired action to take place. Informed *decision-making* is a key feature of the perception-action cycle. It requires the establishment of choices and the selection of one according to a desired goal. There is an element of *prediction* that arises in which perceptual information is combined with a model of the environment to predict the effect actions may have on it. *Attention* is closely related to perception and may be thought of as the requirement to allocate and direct the sensing resources towards relevant information.

Artificial cognition in a cognitive/FAR system attempts to mimic the PA cycle to make the best use of system resources for the situation at hand [26]-[29]. According to Haykin [28], a cognitive radar has the following capabilities: it predicts the consequences of actions, performs explicit decision-making, learns from the environment, and uses memory to store the learned knowledge. Haykin's approach uses Bayesian filtering [87],[88] as the basis for the artificial PA cycle. It provides the processing, perception, and memory inherent in a cognitive system. The Bayesian filter is augmented with a decision-making controller that uses perception from the filter and prediction of future outcomes to determine the next action taken by the sensor.

In [30]-[34], Metron and OSU developed a general mathematical framework for a cognitive/FAR system that incorporates the main components of cognition. The basic mathematical model of a cognitive sensor/processor system is shown in Figure 1. The system consists of four components: (i) the scene, which includes the target and the environment, (ii) the sensor that observes the scene and generally consists of a transmitter and receiver, (iii) the processor that converts the observed data into a perception of the scene, and (iv) the controller that determines the next actions taken by the sensor and processor based on feedback (perception) from the processor. The controller is the novel component that distinguishes a cognitive system from a traditional feed-forward sensor/processor system.

In this framework, we assume that the objective of the system is to estimate the state of a target in the scene. The target state at time t_k is denoted as $\mathbf{x}_k$. The sensor observes the scene and produces a measurement vector $\mathbf{z}_k$ that depends on the target state $\mathbf{x}_k$ and the sensor parameters $\boldsymbol{\theta}_k$. We assume that the estimate of the target state at time t_k is a function of the observations up to time t_k, which in turn depend on the sensor parameters up to time t_k, which we denote as $\mathbf{Z}_k \equiv \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\}$ and $\boldsymbol{\Theta}_k \equiv \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_k\}$, respectively.

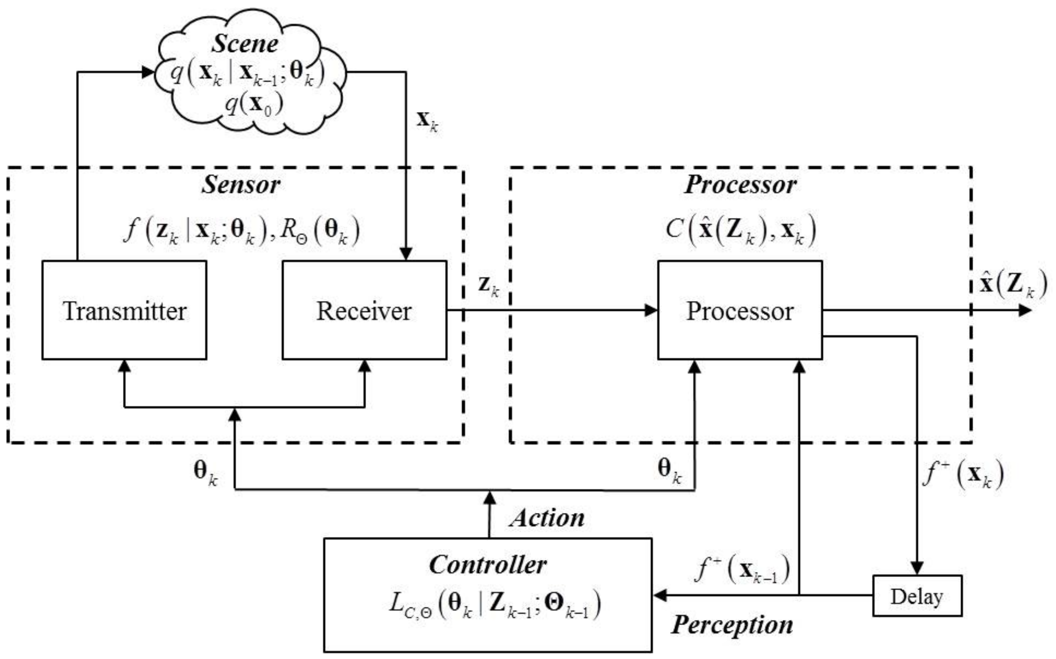

Scene
$q(\mathbf{x}_k \mid \mathbf{x}_{k-1};\boldsymbol{\theta}_k)$
$q(\mathbf{x}_0)$
$\mathbf{x}_k$
Sensor
$f(\mathbf{z}_k \mid \mathbf{x}_k;\boldsymbol{\theta}_k), R_\Theta(\boldsymbol{\theta}_k)$
Transmitter
Receiver
$\mathbf{z}_k$
Processor
$C(\hat{\mathbf{x}}(\mathbf{Z}_k),\mathbf{x}_k)$
Processor
$\hat{\mathbf{x}}(\mathbf{Z}_k)$
$\boldsymbol{\theta}_k$
$\boldsymbol{\theta}_k$
$f^+(\mathbf{x}_k)$
Action
Controller
$L_{C,\Theta}(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1})$
$f^+(\mathbf{x}_{k-1})$
Perception
Delay

Figure 1: Cognitive Sensor/Processor System Framework

We assume a first-order Markov motion model with initial target state probability density function (PDF) $q(\mathbf{x}_0)$ and transition PDF $q(\mathbf{x}_k|\mathbf{x}_{k-1};\boldsymbol{\theta}_k)$, which may depend on the sensor parameters. This will occur, for example, when the choice of sensor parameters affects the time difference t_k-t_{k-1}. The measurement model is described by the conditional PDF $f(\mathbf{z}_k \mid \mathbf{x}_k; \boldsymbol{\theta}_k)$, which is also called the likelihood function. The cost of obtaining an observation and any constraints on the sensor parameters are modeled by the sensor cost function $R_\Theta(\boldsymbol{\theta}_k)$. The processor processes the data and produces an estimate of the target state $\hat{\mathbf{x}}_k(\mathbf{Z}_k)$ by minimizing the expected value of the processor cost function $C(\hat{\mathbf{x}}_k(\mathbf{Z}_k),\mathbf{x}_k)$.

The controller decides on the next value for the sensor parameters $\boldsymbol{\theta}_k$ by minimizing a loss function $L_{C,\Theta}(\cdot)$ that balances the performance of the processor via the processor cost function $C(\cdot,\cdot)$ and the cost of using the sensor via the sensor cost function $R_\Theta(\cdot)$.

For the first-order Markov motion model, the conditional or posterior PDF of $\mathbf{x}_k$ given $\mathbf{Z}_k$ may be obtained from the Bayes-Markov recursion [87]-[89]:

$$f^-(\mathbf{x}_k) \equiv f(\mathbf{x}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_k) = \int q(\mathbf{x}_k \mid \mathbf{x}_{k-1};\boldsymbol{\theta}_k) f^+(\mathbf{x}_{k-1}) d\mathbf{x}_{k-1} \tag{1}$$

$$f^+(\mathbf{x}_k) \equiv f(\mathbf{x}_k \mid \mathbf{Z}_k;\boldsymbol{\Theta}_k) = \frac{f(\mathbf{z}_k \mid \mathbf{x}_k;\boldsymbol{\theta}_k) f^-(\mathbf{x}_k)}{\int f(\mathbf{z}_k \mid \mathbf{x}_k;\boldsymbol{\theta}_k) f^-(\mathbf{x}_k) d\mathbf{x}_k}, \tag{2}$$

where $f^-(\mathbf{x}_k)$ is the motion-updated predicted density and $f^+(\mathbf{x}_k)$ is the information-updated posterior density. The recursion is initialized with

$$f^+(\mathbf{x}_0) = q(\mathbf{x}_0). \tag{3}$$

The *conditional Bayes risk* is the expected value of the processor cost function with respect to the conditional PDF of $\mathbf{x}_k$ given $\mathbf{Z}_k$,

$$R_C^+\left(\mathbf{Z}_k;\mathbf{\Theta}_k\right) = E_k^+\left\{C\left(\hat{\mathbf{x}}\left(\mathbf{Z}_k\right),\mathbf{x}_k\right)\right\}, \tag{4}$$

where $E_k^+\{\cdot\}$ denotes expectation with respect to $f^+\left(\mathbf{x}_k\right)$. The estimator is found by minimizing the conditional Bayes risk [90]:

$$\hat{\mathbf{x}}\left(\mathbf{Z}_k\right) = \underset{\hat{\mathbf{x}}(\mathbf{Z}_k)}{\arg\min}\, R_C^+\left(\mathbf{Z}_k;\mathbf{\Theta}_k\right). \tag{5}$$

In the controller, we assume that we have received the observations up to time t_{k-1} and want to find the next set of sensor parameters to optimize the performance of the state estimator that will include the next observation $\mathbf{z}_k$ as well as the previous observations $\mathbf{Z}_{k-1}$. We define the *joint conditional PDF* of $\mathbf{x}_k$ and $\mathbf{z}_k$ conditioned on $\mathbf{Z}_{k-1}$ as:

$$\begin{aligned} f^{\uparrow}\left(\mathbf{x}_k,\mathbf{z}_k\right) &\equiv f\left(\mathbf{x}_k,\mathbf{z}_k \mid \mathbf{Z}_{k-1};\mathbf{\Theta}_k\right) \\ &= f\left(\mathbf{z}_k \mid \mathbf{x}_k;\boldsymbol{\theta}_k\right) f\left(\mathbf{x}_k \mid \mathbf{Z}_{k-1};\mathbf{\Theta}_k\right) \\ &= f\left(\mathbf{z}_k \mid \mathbf{x}_k;\boldsymbol{\theta}_k\right) f^-\left(\mathbf{x}_k\right). \end{aligned} \tag{6}$$

We define the *predicted conditional Bayes risk* for the estimator $\hat{\mathbf{x}}_k(\mathbf{Z}_k)$ by taking the expectation of the processor cost function with respect to the joint conditional PDF $f^{\uparrow}\left(\mathbf{x}_k,\mathbf{z}_k\right)$,

$$R_C^{\uparrow}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\mathbf{\Theta}_{k-1}\right) \equiv E_k^{\uparrow}\left\{C\left(\hat{\mathbf{x}}\left(\mathbf{Z}_k\right),\mathbf{x}_k\right)\right\}. \tag{7}$$

It is important to emphasize that the predicted conditional Bayes risk is a function of the known past observations $\mathbf{Z}_{k-1}$ but not the unknown next observation $\mathbf{z}_k$ since it has been averaged over both $\mathbf{z}_k$ and $\mathbf{x}_k$. It is also function of all the sensor parameters in $\mathbf{\Theta}_k$, however we separate the dependence on the unknown next sensor parameter $\boldsymbol{\theta}_k$ from the known past sensor parameters $\mathbf{\Theta}_{k-1}$ so that we may optimize over $\boldsymbol{\theta}_k$.

The next value of $\boldsymbol{\theta}_k$ is chosen to minimize a loss function that balances the predicted conditional Bayes risk and the sensor cost,

$$L_{C,\Theta}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\mathbf{\Theta}_{k-1}\right) = L\left\{R_C^{\uparrow}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\mathbf{\Theta}_{k-1}\right), R_{\Theta}\left(\boldsymbol{\theta}_k\right)\right\}. \tag{8}$$

The controller optimization problem is then given by:

$$\boldsymbol{\theta}_k = \underset{\boldsymbol{\theta}}{\arg\min}\; L_{C,\Theta}\left(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\mathbf{\Theta}_{k-1}\right). \tag{9}$$

The cognitive sensor/processor system framework described by Figure 1 and Eqs. (1)-(9) is very general and can be applied to many problems. It is based on the perception-action cycle and includes sensing, processing, perception, memory, attention, prediction, and decision-making. In [30]-[34], the framework was specialized for single target tracking, as well as for target detection and track initiation/termination. A summary of the results is given in Sections 3.1.2-3.1.4. By separating the general principles from the specific application and implementation details, our formulation provides a flexible framework applicable to the general tracking problem. It provides a generalization and formalism to the cognitive radar tracking formulations in [26]-[28],[51]-[65].

3.1.2 Cognitive Single Target Tracking

To specialize the framework for single target tracking, we specify the processor cost function and derive the corresponding state estimator and predicted conditional Bayes risk function used by the controller.

For vector parameter estimation, a commonly used cost function is the sum of the squared estimation errors, which is given by:

$$C\left(\hat{\mathbf{x}}\left(\mathbf{Z}_k\right),\mathbf{x}_k\right)=\mathrm{tr}\left\{\left[\hat{\mathbf{x}}\left(\mathbf{Z}_k\right)-\mathbf{x}_k\right]\left[\hat{\mathbf{x}}\left(\mathbf{Z}_k\right)-\mathbf{x}_k\right]^T\right\}. \qquad (10)$$

For this cost function, the solution to (5) is the minimum mean-square error (MMSE) estimator, which is the conditional mean **Error! Reference source not found.**:

$$\hat{\mathbf{x}}\left(\mathbf{Z}_k\right)=\boldsymbol{\mu}_k^+\equiv E_k^+\left\{\mathbf{x}_k\right\}. \qquad (11)$$

The predicted conditional Bayes risk is given by:

$$R_C^{\uparrow}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}\right)=\mathrm{tr}\left\{\mathbf{S}_k^{\uparrow}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}\right)\right\}, \qquad (12)$$

where

$$\mathbf{S}_k^{\uparrow}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}\right)\equiv E_k^{\uparrow}\left\{\left[\hat{\mathbf{x}}\left(\mathbf{Z}_k\right)-\mathbf{x}_k\right]\left[\hat{\mathbf{x}}\left(\mathbf{Z}_k\right)-\mathbf{x}_k\right]^T\right\} \qquad (13)$$

is the predicted conditional MSE matrix.

In most cases, it is not possible to evaluate the MSE analytically or numerically. However, the *Bayesian Cramér-Rao lower bound* (BCRLB), which is the inverse of the *Bayesian information matrix* (BIM), provides a (matrix) lower bound on the MSE matrix of any estimator [90],[91] and is usually analytically tractable. Here we develop a *predicted conditional BIM* (PC-BIM), $\mathbf{B}_k^{\uparrow}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}\right)$, *and a predicted conditional Cramér-Rao lower bound* (PC-CRLB) to bound the predicted conditional MSE matrix in (13),

$$R_C^{\uparrow}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}\right)\geq \mathrm{tr}\left\{\mathbf{C}^T\mathbf{B}_k^{\uparrow}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}\right)^{-1}\mathbf{C}\right\}. \qquad (14)$$

The PC-BIM may be expressed as the sum of two terms as follows:

$$\mathbf{B}_k^{\uparrow}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}\right)\equiv \mathbf{B}_k^{-}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}\right)+\mathbf{J}_k^{-}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}\right). \qquad (15)$$

The first term is the *predicted information matrix* (PIM), which can be approximated by the inverse of the predicted covariance matrix:

$$\mathbf{B}_k^{-}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}\right)\cong \boldsymbol{\Sigma}_k^{-}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}\right)^{-1}, \qquad (16)$$

and the second term is the expected value of the Fisher information matrix (FIM) with respect to the predicted density $f^-\left(\mathbf{x}_k\right)$. The *expected Fisher information matrix* (EFIM) is given by:

$$\mathbf{J}_k^{-}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}\right)=E_k^{-}\left\{\mathbf{J}\left(\mathbf{x}_k;\boldsymbol{\theta}_k\right)\right\}, \qquad (17)$$

where $\mathbf{J}\left(\mathbf{x}_k;\boldsymbol{\theta}_k\right)$ is the standard FIM [90],[91].

The expressions in (1)-(3), (11), and (14)-(17) provide the Bayes-Markov tracking recursion, the state estimate, and the predicted conditional Bayes risk expressions for a cognitive sensor/processor system whose objective is single target tracking. The cognitive single target tracking system recursion is summarized in Table 1.

Table 1: Cognitive Single Target Tracking Recursion

Initialization	
1	$f^+(\mathbf{x}_0) = q(\mathbf{x}_0)$
Controller Optimization	
2	$f^-(\mathbf{x}_k;\boldsymbol{\theta}) = \int q(\mathbf{x}_k \mid \mathbf{x}_{k-1};\boldsymbol{\theta})\, f^+(\mathbf{x}_{k-1})\, d\mathbf{x}_{k-1}$
3	$\mathbf{B}_k^-(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}) \cong \boldsymbol{\Sigma}_k^-(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1})^{-1}$
4	$\mathbf{J}_k^-(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}) = E_k^-\{\mathbf{J}(\mathbf{x}_k;\boldsymbol{\theta})\}$
5	$\mathbf{B}_k^{\uparrow}(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}) \equiv \mathbf{B}_k^-(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}) + \mathbf{J}_k^-(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1})$
6	$\boldsymbol{\theta}_k = \arg\min_{\boldsymbol{\theta}} \; L\left\{\mathbf{B}_k^{\uparrow}(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}), R_{\Theta}(\boldsymbol{\theta})\right\}$
Motion Update	
7	$f^-(\mathbf{x}_k) = \int q(\mathbf{x}_k \mid \mathbf{x}_{k-1};\boldsymbol{\theta}_k)\, f^+(\mathbf{x}_{k-1})\, d\mathbf{x}_{k-1}$
Measurement	
8	Obtain measurement $\mathbf{z}_k$ according to $\boldsymbol{\theta}_k$
Information Update	
9	$f^+(\mathbf{x}_k) = \dfrac{f(\mathbf{z}_k \mid \mathbf{x}_k;\boldsymbol{\theta}_k)\, f^-(\mathbf{x}_k)}{\int f(\mathbf{z}_k \mid \mathbf{x}_k;\boldsymbol{\theta}_k)\, f^-(\mathbf{x}_k)\, d\mathbf{x}_k}$
Track Estimate	
10	Obtain $\hat{\mathbf{x}}(\mathbf{Z}_k)$ from mean of $f^+(\mathbf{x}_k)$

3.1.3 Cognitive Single Target Tracking, Initiation, and Termination

For track initiation and termination, the system objective is to minimize the time to detect the presence or absence of a target. We do this by maximizing the probability of making a correct decision at each time. To specialize the general cognitive framework for this problem, we follow the likelihood ratio detection and tracking (LRDT) methodology in Chapter 7 of [87]. On H_1 when the target is present, $\mathbf{x}_k$ Î $\mathbf{X}$, where $\mathbf{X}$ is the target-present state space. We define a null (target-absent) state $\varnothing$ so that on H_0 when the target is absent, $\mathbf{x}_k = \varnothing$:

$$\begin{aligned} H_0 &: \mathbf{x}_k = \varnothing \\ H_1 &: \mathbf{x}_k \in \mathbf{X}. \end{aligned} \tag{18}$$

We define the augmented state space $\mathsf{X}^{\varnothing} \equiv \mathsf{X} \cup \varnothing$ and develop the Bayes-Markov recursions for this model. We then specify the processor cost function and derive the corresponding state estimator and predicted conditional Bayes risk function used by the controller.

A PDF on the augmented state space $\mathsf{X}^{\varnothing}$ is characterized by two components $P(\varnothing_k)$ and $f(\mathbf{x}_k)$, where

$$P(\varnothing_k) \equiv \Pr(\mathbf{x}_k = \varnothing), \tag{19}$$

and $f(\mathbf{x}_k)$ is the conditional PDF of $\mathbf{x}_k$ given that $\mathbf{x}_k$ Î X. Therefore, the predicted and posterior PDFs on the augmented state space are characterized by components $P^-(\varnothing_k)$, $f^-(\mathbf{x}_k)$ and $P^+(\varnothing_k)$, $f^+(\mathbf{x}_k)$, respectively.

The initial target PDF is characterized by $P(\varnothing_0)$ and $q(\mathbf{x}_0)$ and the transition density on the augmented state space is characterized by four components:

$$\begin{array}{ll} P(\varnothing_k \mid \varnothing_{k-1}) \equiv \Pr(\mathbf{x}_k = \varnothing \mid \mathbf{x}_{k-1} = \varnothing) & \mathbf{x}_k = \varnothing, \mathbf{x}_{k-1} = \varnothing \\ P(\varnothing_k \mid \mathsf{X}_{k-1}) \equiv \Pr(\mathbf{x}_k = \varnothing \mid \mathbf{x}_{k-1} \in \mathsf{X}) & \mathbf{x}_k = \varnothing, \mathbf{x}_{k-1} \in \mathsf{X} \\ q(\mathbf{x}_k \mid \varnothing_{k-1}) & \mathbf{x}_k \in \mathsf{X}, \mathbf{x}_{k-1} = \varnothing \\ q(\mathbf{x}_k \mid \mathbf{x}_{k-1}; \boldsymbol{\theta}_k) & \mathbf{x}_k \in \mathsf{X}, \mathbf{x}_{k-1} \in \mathsf{X}. \end{array} \tag{20}$$

The likelihood function on the augmented state space is characterized by the likelihood functions on H_0 and H_1, which are denoted by $f(\mathbf{z}_k \mid \varnothing_k; \boldsymbol{\theta}_k)$ and $f(\mathbf{z}_k \mid \mathbf{x}_k; \boldsymbol{\theta}_k)$, respectively.

We express the Bayes-Markov recursions for the predicted and posterior PDFs as recursions on $P^-(\varnothing_k)$, $f^-(\mathbf{x}_k)$, $P^+(\varnothing_k)$, and $f^+(\mathbf{x}_k)$. Following [87], the Bayes-Markov recursions are initialized with:

$$P^+(\varnothing_0) = P(\varnothing_0) \tag{21}$$

$$f^+(\mathbf{x}_0) = q(\mathbf{x}_0). \tag{22}$$

If we assume that the transition probabilities are defined such that the motion update does not affect the probability of being in the null state, then we have:

$$P^-(\varnothing_k) = P^+(\varnothing_{k-1}). \tag{23}$$

Under this assumption, we obtain the "simplified recursion" [87] in which the predicted PDF on H_1 is found from:

$$f^-(\mathbf{x}_k) = P(\varnothing_k \mid \mathsf{X}_{k-1}) q(\mathbf{x}_k \mid \varnothing_{k-1}) + \left[1 - P(\varnothing_k \mid \mathsf{X}_{k-1})\right] \int q(\mathbf{x}_k \mid \mathbf{x}_{k-1}; \boldsymbol{\theta}_k) f^+(\mathbf{x}_{k-1}) d\mathbf{x}_{k-1}. \tag{24}$$

For target detection, we assume that the state estimate takes one of two values, $\hat{\mathbf{x}}_k(\mathbf{Z}_k) = \varnothing$ or $\hat{\mathbf{x}}_k(\mathbf{Z}_k)$Î X, thus the estimation problem becomes a binary detection problem. We assume the standard binary detection cost function [87],[90], and the optimal state estimator (decision rule), found by minimizing the conditional Bayes risk, is the *Bayesian integrated likelihood ratio test* (BLRT) [87]:

$$\bar{\Lambda}\left(\mathbf{Z}_k\right) \underset{\hat{\mathbf{x}}_k = \varnothing}{\overset{\hat{\mathbf{x}}_k \in \mathbf{X}}{\gtrless}} \tau , \tag{25}$$

where $\bar{\Lambda}\left(\mathbf{Z}_k\right)$ is the *Bayesian integrated likelihood ratio* (BLR). For the simplified recursion, it has the form:

$$\bar{\Lambda}\left(\mathbf{Z}_k\right) \equiv \bar{\Lambda}\left(\mathbf{Z}_{k-1}\right) \bar{\mathcal{L}}\left(\mathbf{z}_k \mid \mathbf{Z}_{k-1}\right), \tag{26}$$

where $\bar{\mathcal{L}}\left(\mathbf{z}_k \mid \mathbf{Z}_{k-1}\right)$ is the *integrated likelihood ratio* (ILR) for the current data, defined as:

$$\bar{\mathcal{L}}\left(\mathbf{z}_k \mid \mathbf{Z}_{k-1}\right) \equiv \int \frac{f\left(\mathbf{z}_k \mid \mathbf{x}_k ; \boldsymbol{\theta}_k\right)}{f\left(\mathbf{z}_k \mid \varnothing_k ; \boldsymbol{\theta}_k\right)} f^-\left(\mathbf{x}_k\right) d\mathbf{x}_k . \tag{27}$$

The information update is given by:

$$P^+\left(\varnothing_k\right) = \frac{1}{1+\bar{\Lambda}\left(\mathbf{Z}_k\right)} \tag{28}$$

$$f^+\left(\mathbf{x}_k\right) = \frac{f\left(\mathbf{z}_k \mid \mathbf{x}_k ; \boldsymbol{\theta}_k\right) f^-\left(\mathbf{x}_k\right)}{\int f\left(\mathbf{z}_k \mid \mathbf{x}_k ; \boldsymbol{\theta}_k\right) f^-\left(\mathbf{x}_k\right) d\mathbf{x}_k} . \tag{29}$$

As discussed in [30],[32],[33], the condition in (23) can be satisfied if we set

$$P\left(\varnothing_k \mid \mathbf{X}_{k-1}\right) = \min\left\{\bar{\Lambda}\left(\mathbf{Z}_{k-1}\right)^{-1}, \bar{P}_{\varnothing}\right\}, \tag{30}$$

where $\bar{P}_{\varnothing}$ is a fixed upper limit close to one.

The predicted conditional Bayes risk includes probabilities of missed and false detections, which are generally difficult to calculate, so we must resort to an approximation or surrogate function to perform the controller optimization. As discussed in [30],[32],[33], we take a heuristic approach and use the same criterion that we used for track estimation, namely to minimize the trace of the PC-CRLB, which is the inverse of the PC-BIM. This maximizes the Bayesian information, which is useful for making statistical inferences about $\mathbf{x}_k$ in both the estimation and detection settings. Thus,

$$R_C^{\uparrow}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}\right) \Rightarrow \operatorname{tr}\left\{\mathbf{C}^T \mathbf{B}_k^{\uparrow}\left(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}\right)^{-1} \mathbf{C}\right\}, \tag{31}$$

where we use the notation Þ to denote replacement by a surrogate function.

For track initiation, we initially assume the target is absent and initialize $\bar{\Lambda}\left(\mathbf{Z}_0\right)$ to some small value well below the target-present detection threshold τ_P. If a target is present, then the BLR will grow over time as evidence in favor of H_1 is accumulated. Eventually it will cross the threshold and the target will be declared present. If the target is absent, the BLR will decrease over time. However, if we allow the BLR to decrease below the initial value during periods when there is no target, then when a target appears and the BLR starts to grow, it will take longer to detect. To prevent this, we restrict the BLR to stay at or above the initial value, which we denote as $\bar{\Lambda}_{\min}$.

Once the target is declared present, processing continues in the same manner, except now we assume the target is present and restrict the BLR to stay at or below some maximum value

$\bar{\Lambda}_{\max}$, which is well above the target-absent detection threshold τ_A. To allow some robustness in the system, we set τ_A to be less than τ_P. This way, once the BLR exceeds τ_P and the target is declared present, sufficient evidence in favor of H_0 must accumulate before the BLR drops below τ_A and the target is declared absent again.[1] Since the target is assumed present, we can compute a target state estimate after the information update and we obtain the simultaneous single target tracking and track initiation/termination recursion summarized in Table 2. This recursion reduces to the single target tracking recursion in Table 1 if we always set $P\left(\varnothing_k \mid \mathbf{X}_{k-1}\right) = 0$ on line 5. This recursion is used in the distributed sensor resource allocation examples in [30]-[34].

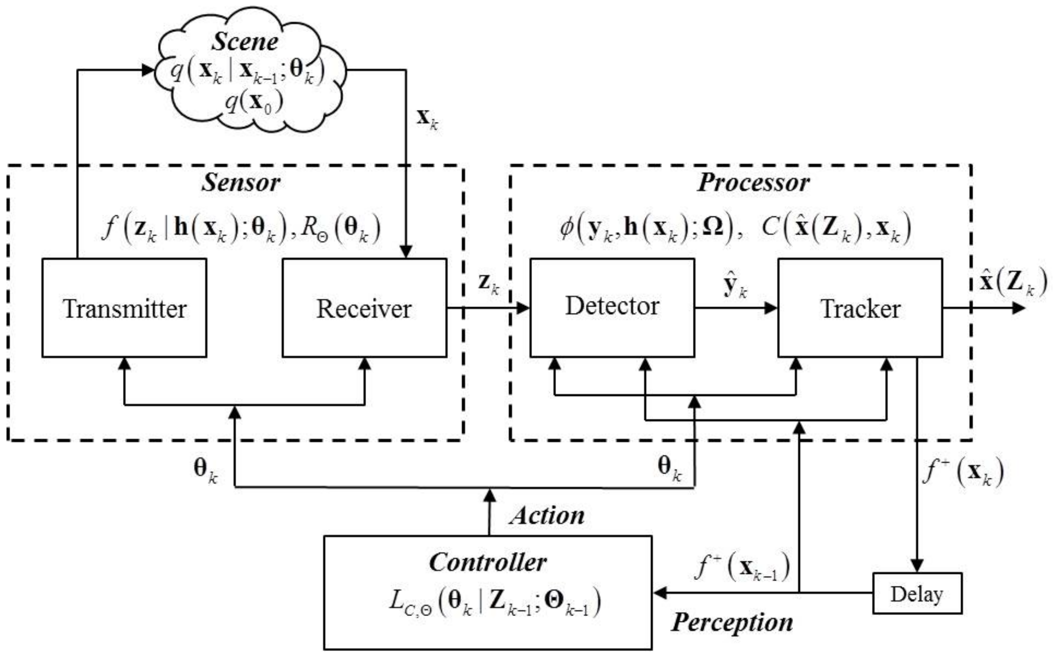
[1] Proper operation of the track initiation and termination recursions requires that t_P £ $\mathrm{L}_{\max}$ and t_A ³ $\mathrm{L}_{\min}$.

Table 2: Cognitive Single Target Tracking, Initiation, and Termination Recursion

```
Initialization
 1   Declare *target absent*
 2   $\bar{\Lambda}(\mathbf{Z}_0) = \bar{\Lambda}_{\min}$
 3   $P^+(\varnothing_0) = \left(1 + \bar{\Lambda}(\mathbf{Z}_0)\right)^{-1}$
 4   $f^+(\mathbf{x}_0) = q(\mathbf{x}_0)$
Motion Update – Part I
 5   $P(\varnothing_k \mid \mathbf{X}_{k-1}) = \min\left\{\bar{\Lambda}(\mathbf{Z}_{k-1})^{-1}, \bar{P}_{\varnothing}\right\}$
 6   $P^-(\varnothing_k) = P^+(\varnothing_{k-1})$
Controller Optimization
 7   $f^-(\mathbf{x}_k;\boldsymbol{\theta}) = P(\varnothing_k \mid \mathbf{X}_{k-1})\, q(\mathbf{x}_k \mid \varnothing_{k-1}) + \left[1 - P(\varnothing_k \mid \mathbf{X}_{k-1})\right] \int q(\mathbf{x}_k \mid \mathbf{x}_{k-1};\boldsymbol{\theta})\, f^+(\mathbf{x}_{k-1})\, d\mathbf{x}_{k-1}$
 8   $\mathbf{B}_k^-(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}) \cong \boldsymbol{\Sigma}_k^-(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1})^{-1}$
 9   $\mathbf{J}_k^-(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}) = E_k^-\left\{\mathbf{J}(\mathbf{x}_k;\boldsymbol{\theta})\right\}$
10   $\mathbf{B}_k^{\uparrow}(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}) \equiv \mathbf{B}_k^-(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}) + \mathbf{J}_k^-(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1})$
11   $\boldsymbol{\theta}_k = \arg\min_{\boldsymbol{\theta}}\ L\left\{\mathbf{B}_k^{\uparrow}(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}), R_{\Theta}(\boldsymbol{\theta})\right\}$
Motion Update – Part II
12   $f^-(\mathbf{x}_k) = P(\varnothing_k \mid \mathbf{X}_{k-1})\, q(\mathbf{x}_k \mid \varnothing_{k-1}) + \left[1 - P(\varnothing_k \mid \mathbf{X}_{k-1})\right] \int q(\mathbf{x}_k \mid \mathbf{x}_{k-1};\boldsymbol{\theta}_k)\, f^+(\mathbf{x}_{k-1})\, d\mathbf{x}_{k-1}$
Measurement
13   Obtain measurement $\mathbf{z}_k$ according to $\boldsymbol{\theta}_k$
BLR
11   $\bar{\mathcal{L}}(\mathbf{z}_k \mid \mathbf{Z}_{k-1}) \equiv \int \frac{f(\mathbf{z}_k \mid \mathbf{x}_k;\boldsymbol{\theta}_k)}{f(\mathbf{z}_k \mid \varnothing_k;\boldsymbol{\theta}_k)} f^-(\mathbf{x}_k)\, d\mathbf{x}_k$
12   $\bar{\Lambda}(\mathbf{Z}_k) = \max\left\{\bar{\Lambda}_{\min}, \min\left\{\bar{\Lambda}_{\max}, \bar{\Lambda}(\mathbf{Z}_{k-1})\, \bar{\mathcal{L}}(\mathbf{z}_k \mid \mathbf{Z}_{k-1})\right\}\right\}$
Information Update
13   $P^+(\varnothing_k) = \left(1 + \bar{\Lambda}(\mathbf{Z}_k)\right)^{-1}$
14   $f^+(\mathbf{x}_k) = \frac{f(\mathbf{z}_k \mid \mathbf{x}_k;\boldsymbol{\theta}_k)\, f^-(\mathbf{x}_k)}{\int f(\mathbf{z}_k \mid \mathbf{x}_k;\boldsymbol{\theta}_k)\, f^-(\mathbf{x}_k)\, d\mathbf{x}_k}$
BLRT
16   **if** *target absent* and $\bar{\Lambda}(\mathbf{Z}_k) \geq \tau_P$, **then** declare *target present* and initiate track,
     **elseif** *target present* and $\bar{\Lambda}(\mathbf{Z}_k) < \tau_A$, **then** declare *target absent* and terminate track
     **end**
Track Estimate (if *target present*)
15   Obtain $\hat{\mathbf{x}}(\mathbf{Z}_k)$ from mean of $f^+(\mathbf{x}_k)$
```

3.1.4 Cognitive MAP-PF Single Target Tracking

The recursions in Tables 1 and 2 are applicable to detection-based and track-before-detect tracking systems. In a detection-based system, the sensor performs "hard detection," i.e. there is some signal processing of the sensor data that converts the data to a detection surface, which is then thresholded to produce a "detection" in the form of a measurement in the natural measurement space of the sensor (e.g. angle, range, and/or Doppler). The observation $\mathbf{z}_k$ is the measurement obtained from hard detection processing. In a track-before-detect system, $\mathbf{z}_k$ is the sensor data and the information update involves computing the likelihood function of the sensor data with respect to the target state vector. This can be computationally intensive but can yield significant performance improvements over a detection-based system. In this subsection, we extend the recursions to maximum a posteriori penalty function (MAP-PF) tracking systems. The MAP-PF methodology applies to problems in which the likelihood function $f\left(\mathbf{z}_k \mid \mathbf{x}_k;\boldsymbol{\theta}_k\right)$ depends on the target state vector only through a known, possibly nonlinear mapping to the natural measurement space of the sensor. The MAP-PF methodology offers reduced computational complexity over track-before-detect systems, while maintaining the performance advantage over detection-based systems. The cognitive MAP-PF tracking system, in which the processor includes the detector and tracker, is shown in Figure 2. It is particularly suitable for FAR systems, as it already contains feedback within the tracking processor.



Scene
$q(\mathbf{x}_k \mid \mathbf{x}_{k-1};\boldsymbol{\theta}_k)$
$q(\mathbf{x}_0)$
$\mathbf{x}_k$
Sensor
$f(\mathbf{z}_k \mid \mathbf{h}(\mathbf{x}_k);\boldsymbol{\theta}_k), R_\Theta(\boldsymbol{\theta}_k)$
Transmitter
Receiver
$\mathbf{z}_k$
Processor
$\phi(\mathbf{y}_k,\mathbf{h}(\mathbf{x}_k);\boldsymbol{\Omega}),\ C(\hat{\mathbf{x}}(\mathbf{Z}_k),\mathbf{x}_k)$
Detector
$\hat{\mathbf{y}}_k$
Tracker
$\hat{\mathbf{x}}(\mathbf{Z}_k)$
$\boldsymbol{\theta}_k$
$\boldsymbol{\theta}_k$
$f^+(\mathbf{x}_k)$
Action
Controller
$L_{C,\Theta}(\boldsymbol{\theta}_k \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1})$
$f^+(\mathbf{x}_{k-1})$
Perception
Delay

Figure 2: Cognitive MAP-PF Sensor/Processor System Framework

MAP-PF is a multi-target tracking methodology developed in [92]-[97] and described in Chapter 6 of [87]. In this approach, the multi-target track estimation problem is formulated directly from the sensor data $\mathbf{z}_k$ using the maximum a posteriori (MAP) estimation criterion. The penalty function method of nonlinear programming [98] is used to obtain a tractable solution. The result is a two-step estimation process similar to traditional feed-forward detection-based systems, except the processes are coupled via the penalty function and the data association step of traditional multi-target tracking approaches is eliminated. In the detection process, the penalty function uses the current target states to guide the detector to the relevant region of the detection surface. In the track estimation process, the penalty function determines the influence of the detector measurements on the final track estimates by adaptively adjusting the measurement error variance using the FIM.

Let $\mathbf{y}_k$ denote the natural parameters. They are related to the state parameters by the nonlinear mapping

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k). \tag{32}$$

Let $f(\mathbf{z}_k \mid \mathbf{y}_k; \boldsymbol{\theta}_k)$ denote the likelihood function with respect to the natural parameters and let $\mathbf{J}_\mathbf{y}(\mathbf{y}_k; \boldsymbol{\theta}_k)$ denote the FIM of the natural parameters. The likelihood function and FIM with respect to the state parameters are given by [90]:

$$f(\mathbf{z}_k \mid \mathbf{x}_k; \boldsymbol{\theta}_k) = f(\mathbf{z}_k \mid \mathbf{y}_k = \mathbf{h}(\mathbf{x}_k); \boldsymbol{\theta}_k) \tag{33}$$

$$\mathbf{J}_\mathbf{x}(\mathbf{x}_k; \boldsymbol{\theta}_k) = \mathbf{H}(\mathbf{x}_k)^T \mathbf{J}_\mathbf{y}\left(\mathbf{h}(\mathbf{x}_k); \boldsymbol{\theta}_k\right) \mathbf{H}(\mathbf{x}_k), \tag{34}$$

where $\mathbf{H}(\mathbf{x}_k)$ is the Jacobian matrix, defined as:

$$\mathbf{H}(\mathbf{x}_k) \equiv \left[\nabla_\mathbf{x} \mathbf{h}(\mathbf{x})^T\right]^T \Big|_{\mathbf{x}=\mathbf{x}_k}. \tag{35}$$

The MAP-PF algorithm employs a penalty function $\phi(\mathbf{y}, \mathbf{h}(\mathbf{x}))$ which is equal to zero when $\mathbf{y} = \mathbf{h}(\mathbf{x})$ and becomes smaller (more negative) as the distance between $\mathbf{y}$ and $\mathbf{h}(\mathbf{x})$ increases. For example, a quadratic penalty function is:

$$\phi(\mathbf{y}, \mathbf{h}(\mathbf{x}); \boldsymbol{\Omega}) \equiv -\frac{1}{2}[\mathbf{y} - \mathbf{h}(\mathbf{x})]^T \boldsymbol{\Omega}^{-1} [\mathbf{y} - \mathbf{h}(\mathbf{x})], \tag{36}$$

where $\boldsymbol{\Omega}$ is a matrix chosen to weight the components of the penalty function in some desirable manner. The MAP-PF motion update is the same as in the standard Bayes-Markov recursion and the MAP-PF information update is given by:

$$\hat{\mathbf{x}}_k^- = \arg\max_{\mathbf{x}_k} f^-(\mathbf{x}_k) \tag{37}$$

$$\hat{\mathbf{y}}_k = \arg\max_{\mathbf{y}} \ln f(\mathbf{z}_k | \mathbf{y}; \boldsymbol{\theta}_k) + \phi\left(\mathbf{y}, \mathbf{h}(\hat{\mathbf{x}}_k^-); \boldsymbol{\Omega}\right) \tag{38}$$

$$f^+(\mathbf{x}_k) = \frac{\exp\left\{\phi\left(\hat{\mathbf{y}}_k, \mathbf{h}(\mathbf{x}_k); \boldsymbol{\Omega}\right)\right\} f^-(\mathbf{x}_k)}{\int \exp\left\{\phi\left(\hat{\mathbf{y}}_k, \mathbf{h}(\mathbf{x}_k); \boldsymbol{\Omega}\right)\right\} f^-(\mathbf{x}_k) \, d\mathbf{x}_k}. \tag{39}$$

In the first step, the MAP estimate of the predicted density is found. Depending on the implementation, it may be easier to find the MMSE estimate, which is the mean of the predicted density, instead.

In the second step, the optimization problem in (38) is a penalized maximum likelihood (ML) problem. If the second term in (38) had the form $\ln f(\mathbf{y})$, it would be a MAP estimation problem. Thus the penalty function can be interpreted as a prior term in a MAP estimation problem. In a traditional feed-forward detection-based tracking system, the optimal detector would solve the standard ML problem (i.e. (38) without the penalty function) to get the detector measurement. In MAP-PF, the penalty function restricts the detector estimate to be in the vicinity of where the tracker predicts it to be, hence MAP-PF is performing "guided" detection. By specifying a quadratic penalty function, we are implicitly modeling the prior distribution of $\mathbf{y}_k$ as Gaussian with mean $\mathbf{h}\left(\hat{\mathbf{x}}_k^-\right)$ and covariance matrix $\mathbf{\Omega}$. We have some flexibility in choosing $\mathbf{\Omega}$, and a logical choice would be the covariance matrix of the predicted density of $\mathbf{y}_k$ obtained from a transformation of the predicted density $f^-\left(\mathbf{x}_k\right)$. Using a locally linear approximation of the function $\mathbf{h}\left(\mathbf{x}\right)$ at the point $\hat{\mathbf{x}}_k^-$, we choose $\mathbf{\Omega}$ to be the predicted covariance matrix of $\mathbf{y}_k$, which is approximately given by:

$$\mathbf{\Omega}_{MAP,k} = \mathbf{\Sigma}_{\mathbf{y}_k}^- \cong \mathbf{H}\left(\hat{\mathbf{x}}_k^-\right)\mathbf{\Sigma}_k^-\mathbf{H}\left(\hat{\mathbf{x}}_k^-\right)^T. \tag{40}$$

The third step in (39) looks like a standard information update with $\hat{\mathbf{y}}_k$ acting as the measurement vector and the exponential of the penalty function, $\exp\left\{\phi\left(\hat{\mathbf{y}}_k, \mathbf{h}\left(\mathbf{x}_k\right); \mathbf{\Omega}\right)\right\}$, acting as the measurement likelihood function $f(\hat{\mathbf{y}}_k \mid \mathbf{x}_k; \boldsymbol{\theta}_k)$. Here the quadratic penalty function is implicitly modeling $\hat{\mathbf{y}}_k$ as Gaussian with mean $\mathbf{h}(\mathbf{x}_k)$ and covariance matrix $\mathbf{\Omega}$**.** As in [87],[92]-[97], we choose $\mathbf{\Omega}$ to be the inverse of the FIM of the natural parameters, $\mathbf{J}_{\mathbf{y}}\left(\mathbf{y}_k; \boldsymbol{\theta}_k\right)$. Calculation of the FIM often requires knowledge of the true value of $\mathbf{y}_k$, however we can obtain a reasonably accurate approximation to the FIM by substituting in an estimate of $\mathbf{y}_k$. The transformation of the predicted state estimate $\mathbf{h}\left(\hat{\mathbf{x}}_k^-\right)$ is a less volatile estimate than the current measurement $\hat{\mathbf{y}}_k$, therefore we evaluate the FIM at $\mathbf{h}\left(\hat{\mathbf{x}}_k^-\right)$. Thus, for the information update we choose

$$\mathbf{\Omega}_{I,k}^{-1} = \mathbf{J}_{\mathbf{y}}\left(\mathbf{h}\left(\hat{\mathbf{x}}_k^-\right); \boldsymbol{\theta}_k\right). \tag{41}$$

The MAP-PF single target tracking recursion is summarized in Table 3. We also developed a MAP-PF single target tracking, initiation, and termination recursion in [30]. The recursion in Table 3 is used in the SDR and CREW examples in [30],[34]-[40].

Table 3: Cognitive Single Target MAP-PF Tracking Recursion

```
Initialization
  1    $f^{+}(\mathbf{x}_0) = q(\mathbf{x}_0)$
Controller Optimization
  2    $f^{-}(\mathbf{x}_k;\boldsymbol{\theta}) = \int q(\mathbf{x}_k \mid \mathbf{x}_{k-1};\boldsymbol{\theta})\, f^{+}(\mathbf{x}_{k-1})\, d\mathbf{x}_{k-1}$
  3    $\mathbf{B}_k^{-}(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}) \cong \boldsymbol{\Sigma}_k^{-}(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1})^{-1}$
  4    $\mathbf{J}_k^{-}(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}) = E_k^{-}\{\mathbf{J}(\mathbf{x}_k;\boldsymbol{\theta})\}$
  5    $\mathbf{B}_k^{\uparrow}(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}) \equiv \mathbf{B}_k^{-}(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}) + \mathbf{J}_k^{-}(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1})$
  6    $\boldsymbol{\theta}_k = \arg\min_{\boldsymbol{\theta}} \; L\{\mathbf{B}_k^{\uparrow}(\boldsymbol{\theta} \mid \mathbf{Z}_{k-1};\boldsymbol{\Theta}_{k-1}), R_{\Theta}(\boldsymbol{\theta})\}$
Motion Update
  7    $f^{-}(\mathbf{x}_k) = \int q(\mathbf{x}_k \mid \mathbf{x}_{k-1};\boldsymbol{\theta}_k)\, f^{+}(\mathbf{x}_{k-1})\, d\mathbf{x}_{k-1}$
Measurement
  8    Obtain measurement $\mathbf{z}_k$ according to $\boldsymbol{\theta}_k$
  9    Obtain $\hat{\mathbf{x}}_k^{-}$ from mean or maximum of $f^{-}(\mathbf{x}_k)$
  10   $\boldsymbol{\Omega}_{MAP,k} = \mathbf{H}(\hat{\mathbf{x}}_k^{-})\boldsymbol{\Sigma}_k^{-}\mathbf{H}(\hat{\mathbf{x}}_k^{-})^{T}$
  11   $\hat{\mathbf{y}}_k = \arg\max_{\mathbf{y}} \; \ln f(\mathbf{z}_k \mid \mathbf{y};\boldsymbol{\theta}_k) + \phi(\mathbf{y}, \mathbf{h}(\hat{\mathbf{x}}_k^{-});\boldsymbol{\Omega}_{LF,k})$
Information Update
  12   $\boldsymbol{\Omega}_{I,k}^{-1} = \mathbf{J}_{\mathbf{y}}(\mathbf{h}(\hat{\mathbf{x}}_k^{-});\boldsymbol{\theta}_k)$
  13   $f^{+}(\mathbf{x}_k) = \dfrac{\exp\{\phi(\hat{\mathbf{y}}_k, \mathbf{h}(\mathbf{x}_k);\boldsymbol{\Omega}_{I,k})\}\, f^{-}(\mathbf{x}_k)}{\int \exp\{\phi(\hat{\mathbf{y}}_k, \mathbf{h}(\mathbf{x}_k);\boldsymbol{\Omega}_{I,k})\}\, f^{-}(\mathbf{x}_k)\, d\mathbf{x}_k}$
Track Estimate
  14   Obtain $\hat{\mathbf{x}}(\mathbf{Z}_k)$ from mean or maximum of $f^{+}(\mathbf{x}_k)$
```

3.2 Examples

For a specific application, we need to specify the components of the state vector, the motion and measurement models, the sensor parameters being controlled, and the form of the controller loss function. Finally, we need to specify the implementation details that include the type of tracker used to implement the Bayes-Markov recursion and the method for solving the controller optimization problem. In [30]-[34], we showed how the general tracking framework could be specialized for a distributed sensor system similar to the cognitive radar networks in [64],[65], in which system resources (observation time on each sensor) were allocated to optimize tracking performance. Using simulated data, we showed that the cognitive radar system offered significant performance gains over a standard feed-forward radar system. In [30],[34],[35], we showed how the tracking framework could be applied to a single sensor pulse-Doppler radar system in which the pulse repetition frequency (PRF) is adjusted to optimize tracking performance, while keeping the target from being Doppler-aliased and away from the zero-Doppler clutter. Results were shown on experimentally collected data using OSU's SDR system. In [36]-[39], we applied the same algorithm to real-time data using OSU's CREW system, and also allowed for simultaneous adjustment of the PRF and number of pulses.

3.3 Summary

In this section, we provided an overview of the general framework for a cognitive sensor/processor tracking system developed in [30]-[34]. The framework is based on the *perception-action cycle* and includes *sensing* in the sensor; *processing* in the detector and tracker; *perception* in the conversion of sensor data to the posterior PDF of the state vector; *memory* of all the past data in the posterior PDF; *attention* in the penalty function of the guided adaptive detector, which focuses the detector on the relevant region of the detection surface; *prediction* in the PC-BIM, which predicts the performance of the next measurement; and *decision-making* in the controller, which decides on the next values for the sensor parameters based on the predicted performance.

4 RESULTS AND DISCUSSION

4.1 FAR M&S Software Architecture Overview

The FAR M&S codebase has been designed in an object-oriented architecture so that the interfaces for the various components (optimizer, sensor, processor, etc.) are defined by base classes. Specific implementations of these objects (for example, an optimizer for five parameters) are subclasses of their respective base class so that they inherit common methods and properties.

The FAR M&S architecture is managed by a *FAR_Engine*. A conceptual block diagram of the FAR_Engine showing the basic objects, processing, and data flow is shown in Figure 3. The FAR_Engine consists of eight objects:

1. *Scene* [Simulation only]: defines the target, clutter and noise characteristics used for simulation.
2. *Optimizer*: solves the controller optimization problem to obtain the next set of sensor parameters
3. *Sensor*: obtains raw data from simulation, pre-recorded data, or experimentation.
4. *Processor*: performs raw data processing and Bayesian filtering.
5. *StorageManager*: maintains a history of variables of interest from the Scene, Optimizer, Sensor, and Processor and stores them to a file (long-term memory).
6. *DisplayManager*: displays values of interest each cycle while the algorithm is running and provides a final display of quantities of interest.
7. *TimingManager*: maintains timing during the cycle.
8. *PerceptionActionCycle*: runs one cycle of the PA cycle by calling object methods (functions) in the proper sequence, as shown in Figure 3.

The repository where the codebase is stored has been divided into two folders: an "architecture" folder to store the base classes and other common pieces of code, and a "modules" folder which stores the specific subclasses or implementations. The "architecture" folder contains eight files defining the base classes plus a *UtilityFunctions* object, which contains some commonly used functions. The FAR Engine is written as a script and there is no base class. The files in the "architecture" folder provide the interfaces (that is, common properties and methods) for how to create specific objects that provide functionality to different parts of the PerceptionActionCycle and FAR_Engine. The files in this folder should not be modified.

In Phase I, we developed specific implementations of the FAR M&S architecture for three examples: (i) the distributed sensor resource allocation (DSRA) simulation example in [33], (ii) the SDR pre-recorded data example in [35], and (iii) a CREW example in which five parameters are optimized and the data may be from simulation or real time experimentation. The "modules" folder contains separate folders for each of these examples named "DSRA," "SDR," and "CREW."

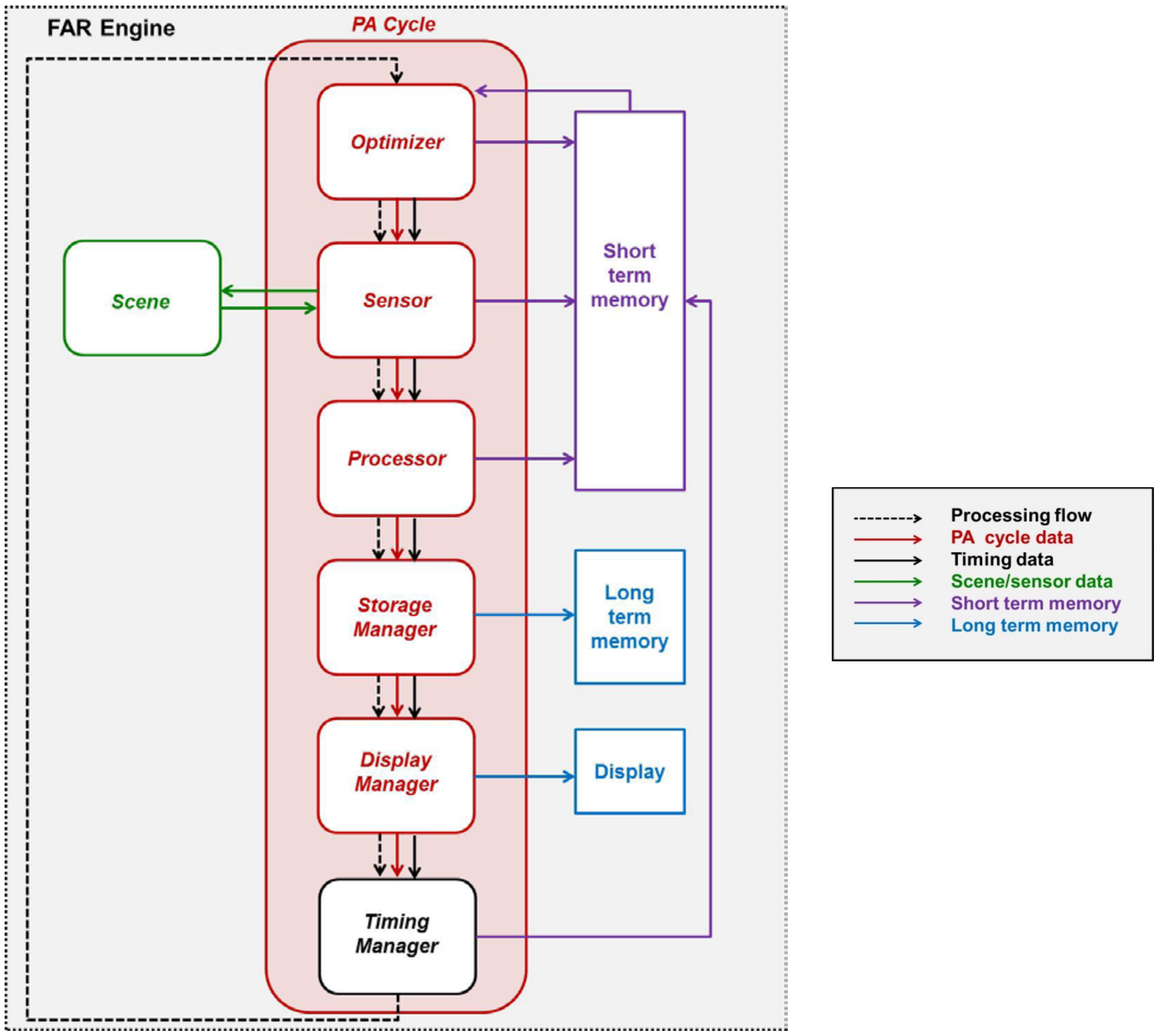


Figure 3: FAR M&S Architecture Block Diagram

4.2 Using the FAR M&S MATLAB Code

4.2.1 System Requirements

To run the codebase, you will need MATLAB version 2014b or later. It is possible that it will run on earlier versions, but this has not been tested. No MATLAB Toolboxes are required.

4.2.2 Getting Started

To get up and running, do the following:

- Install the codebase into a folder on your computer.
- Start MATLAB.
- In MATLAB, navigate to the folder that you just stored. In this folder there should at least be a "code" folder and a file named `set_matlab_path_far_codebase()`.
- At the command prompt, type `set_matlab_path_far_codebase()`. This sets the proper paths for use with the code.

You can now run any of the three examples provided in the codebase. Navigate to one of the folders "code/modules/DSRA", "code/modules/SDR", or "code/modules/CREW". Open the FAR_Engine file for that example (`FAR_Engine_JSTSP`, `FAR_Engine_SDR_PRF`, or `FAR_Engine_CREW_5Par`) and run it.

4.3 FAR M&S Architecture Base Classes

In this section, we provide an overview of the base classes in the FAR M&S Architecture. For the Scene, Sensor, Processor, Optimizer, Storage Manager, Display Manager, and Timing Manager, we provide a list of properties and abstract methods in each class. The PerceptionActionCycle class implements a method to run the cycle and it is described in detail.

The DSRA, SDR, and CREW modules provide implementations of each of the base classes and the FAR_Engines to run the examples for those applications. In Section 4.4, we provide a description of the CREW Sensor and Scene objects. We do not provide descriptions of the remaining codebase modules.

4.3.1 Scene

The Scene base class consists of

- Properties: N/A
- Abstract Methods: N/A

This is currently just a placeholder as we have not identified properties or methods that are common to all Scenes.

4.3.2 Sensor

The Sensor base class consists of:

- Properties: N/A
- Abstract Methods: get_measurement

4.3.3 Processor

The Processor base class consists of:

- Properties: MotionModel (structure), MeasModel (structure), Settings (structure)
- Abstract Methods: preprocess_rawdata, prior_initialization, predicted_update, posterior_update

4.3.4 Optimizer

The Optimizer base class consists of:

- Properties: PredActualFlag
- Abstract Methods: get_sensor_parameters, set_sensor_params, opt_sensor_params_pred_update

4.3.5 Storage Manager

The StorageManager base class consists of:

- Properties: Params (structure), history (structure)
- Abstract Methods: history_update

4.3.6 Display Manager

The DisplayManagerManager base class consists of:

- Properties: Params (structure), DisplayData (structure)
- Abstract Methods: store_display_data, display_cycle

4.3.7 Timing Manager

The TimingManager base class consists of:

- Properties: time (structure), k
- Abstract Methods: cycle_timing_init, cycle_timing_opt, cycle_timing_sens, cycle_timing_proc, cycle_timing_mgmt, timing_update

4.3.8 Perception Action Cycle

The PerceptionActionCycle base class consists of:

- Properties: f_post (structure), theta_opt (structure), cycle_timing (structure)
- Methods: PerceptionActionCycle (constructor), run_cycle

The PerceptionActionCycle base class provides implementations for the object constructor method and for the run_cycle method. Thus, the PerceptionActionCycle object is common to all examples.

The properties f_post (the posterior density), theta_opt (the optimum sensor parameters for the current cycle), and cycle_timing (various timing values for the current cycle) are updated during the execution of run_cycle and are passed out to the FAR_Engine at the end of the cycle. They are passed back in to run_cycle at the beginning of the next cycle. Passing these parameters to the next cycle is represented by the purple short term memory data in Figure 3.

The constructor method PerceptionActionCycle(f_post, theta0) is called by FAR_Engine up front to create the PerceptionActionCycle object and initialize the posterior density and sensor parameters. The code is

```
function cycleObj = PerceptionActionCycle(f_post, theta0)

     if nargin >0
         cycleObj.f_post       = f_post;
         cycleObj.theta_opt    = theta0;
         cycleObj.cycle_timing = [];
     end % if nargin

 end % constructor function
```

The run_cycle method implements the PA cycle. A detailed description is provided below.

```
function [cycleObj,sensorObj,processorObj,timingObj,storageObj,displayObj]...
        = run_cycle(cycleObj,sceneObj, sensorObj, processorObj,
        optimizerObj, timingObj, storageObj, displayObj)
```

This defines the interface for calling run_cycle. *Inputs are the scene, sensor, processor, optimizer, timing, storage, and display objects. During* run_cycle, *the cycle, sensor, processor, timing, storage, and display objects are updated and passed out to the FAR_Engine.*

```
    %-----------------------------------------------------------------------
    % previous sensor parameters
    %-----------------------------------------------------------------------
    theta_old = cycleObj.theta_opt;
```

This gets the optimum sensor values from the last cycle.

```
    %-----------------------------------------------------------------------
    % initialize cycle timing
    %-----------------------------------------------------------------------
    cycleObj.cycle_timing = timingObj.cycle_timing_init;
```

This calls the method `cycle_timing_init`, *implemented in* `timingObj`, *a specific implementation of the* `TimingMagager` *object. It initializes the timing values for the current cycle.*

```
    %-----------------------------------------------------------------------
    % tentative motion update & theta optimization
    %-----------------------------------------------------------------------
    [cycleObj.theta_opt, optParams, f_pred] = ...
            optimizerObj.opt_sensor_params_pred_update(...
            theta_old, cycleObj.f_post, sensorObj, processorObj,...
            cycleObj.cycle_timing);
```

This calls the method `opt_sensor_params_pred_update`, *implemented in* `optimizerObj`, *a specific implementation of the* `Optimizer` *object. It performs the controller optimization to obtain the next set of sensor parameters. Inputs are the previous set of sensor values, the posterior density from last cycle, the sensor and processor objects, and the cycle_timing structure. As part of the optimization, the predicted density is computed for each set of sensor values that is evaluated. In some cases, the actual predicted density is computed and returned to the PA cycle in the variable* `f_pred`. *In this case the property* `optimizerObj.PredActualFlag` *is set equal to true. If not, it is false. The optimum sensor parameters are returned in* `cycleObj.theta_opt` *and other quantities of interest calculated during the optimization are returned in* `optParams`. *These will be stored or displayed later.*

```
    cycleObj.cycle_timing = timingObj.cycle_timing_opt(optimizerObj,...
            cycleObj.cycle_timing);
```

This calls the method `cycle_timing_opt`, *implemented in* `timingObj`. *It stores the optimization timing values for the current cycle. Inputs are the optimizer object and the cycle timing structure, and the output is the updated cycle timing structure.*

```
%------------------------------------------------------------------
% set sensor/processor parameters and get measurement
%------------------------------------------------------------------
[sensorObj, processorObj] = optimizerObj.set_sensor_params(...
        cycleObj.theta_opt, sensorObj, processorObj);
```

This calls the method `set_sensor_params`*, implemented in* `optimizerObj`*. It updates the sensor and processor objects with the new sensor parameter values. Inputs are the optimum sensor values and the sensor and processor objects. Outputs are the updated sensor and processor objects.*

```
[rawdata, sens_timing] = sensorObj.get_measurement(sceneObj,...
        cycleObj.cycle_timing);
```

This calls the method `get_measurement`*, implemented in* `sensorObj`*, a specific implementation of the* `Sensor` *object. It gets the measurement for the appropriate data source. Inputs are the scene object (if simulation), and timing for the data collect. Outputs are the raw sensor data and sensor timing values.*

```
cycleObj.cycle_timing = timingObj.cycle_timing_sens(sens_timing,...
        cycleObj.cycle_timing);
```

This calls the method `cycle_timing_sens`*, implemented in* `timingObj`*. It stores the sensor timing values for the current cycle. Inputs are the sensor timing structure and the cycle timing structure, and the output is the updated cycle timing structure. One of the values is a flag that indicates if data was available. The following loop to process the data is only executed if data is available.*

```
if cycleObj.cycle_timing.data_available
%------------------------------------------------------------------
% actual motion update
%------------------------------------------------------------------
    if ~optimizerObj.PredActualFlag
        f_pred = processorObj.predicted_update(cycleObj.f_post, ...
                cycleObj.cycle_timing.dt, cycleObj.theta_opt, theta_old);
    end
```

This calls the method `predicted_update`*, implemented in* `processorObj`*, a specific implementation of the* `Processor` *object. It performs the Bayesian filtering motion update, if not already performed as part of the sensor optimization. Inputs are the posterior density from last cycle, the total scan time of the current cycle, and the current and previous values of the sensor parameters. The output is the predicted density.*

```
    %------------------------------------------------------------------
    % pre-process raw data
    %------------------------------------------------------------------
    zdata = processorObj.preprocess_rawdata(rawdata, sensorObj);
```

This calls the method `process_rawdata`*, implemented in* `processorObj`*. It performs pre-processing of the raw data to get it into the form required by the information update, such as* *converting raw radar data to a range/Doppler surface. Inputs are the raw data and the sensor object. The output is the processed data.*

```
        %-------------------------------------------------------------------
        % information update
        %-------------------------------------------------------------------
        [cycleObj.f_post, LF, proc_timing] = ...
                processorObj.posterior_update(f_pred, zdata);
```

This calls the method `posterior_update`*, implemented in* `processorObj`*. It performs the information update. Inputs are the predicted density and processed data. Outputs are the posterior density, the likelihood function structure containing quantities of interest to be stored or displayed later, and processor timing values.*

```
        else
            f_pred = cycleObj.f_post;
            LF = [];
            proc_timing.track_new = 0;
        end  % if data_available
```

If there is no data available, the predicted density is the posterior density from the last cycle, and the posterior density is also the posterior density from last cycle. The likelihood structure is empty and the flag indicating a new track is false.

```
    cycleObj.cycle_timing = timingObj.cycle_timing_proc(proc_timing,...
            cycleObj.cycle_timing);
```

This calls the method `cycle_timing_proc`*, implemented in* `timingObj`*. It stores the processor timing values for the current cycle. Inputs are the processor timing structure and the cycle timing structure, and the output is the updated cycle timing structure.*

```
    %-------------------------------------------------------------------
    % storage update
    %-------------------------------------------------------------------
    storageObj = storageObj.history_update(cycleObj.cycle_timing,...
            cycleObj.theta_opt, optParams, f_pred, cycleObj.f_post, LF);
```

This calls the method `history_update`*, implemented in* `storageObj`*, a specific implementation of the* `StorageManager` *object. It stores quantities of interest from the current cycle in the* `history` *structure, to be saved to a file later. Inputs are the cycle timing, the optimum sensor parameters, the additional optimization values, the predicted density, the posterior density, and the likelihood function structure. The output is the updated storage object.*

```
    %-------------------------------------------------------------------
    % display update
    %-------------------------------------------------------------------
    if displayObj.Params.PlotCycleFlag
        displayObj = displayObj.store_display_data(cycleObj.cycle_timing,...
                cycleObj.theta_opt, optParams, f_pred, cycleObj.f_post, LF);
        displayObj.display_cycle(timingObj.k);
    end
```

There is an option not to display any data while the algorithm is running in order to save time. If the `displayObj.Params.PlotCycleFlag` *is false, then nothing is done. If it is true, then this code calls the methods* `store_display_data` *and* `display_cycle`*, implemented in* `displayObj`*, a specific implementation of the* `DisplayManager` *object. The method* `store_display_data` *stores quantities of interest from the current cycle in the display object and* `display_cycle` *updates the display screen. Inputs to* `store_display_data` *are the cycle timing, the optimum sensor parameters, the additional optimization values, the predicted density, the posterior density, and the likelihood function structure. The output is the updated display object. The inputs to* `display_cycle` *are the current cycle index and (implicitly) the display object.*

```
    cycleObj.cycle_timing = timingObj.cycle_timing_mgmt(...
            cycleObj.cycle_timing);
```

This calls the method `cycle_timing_mgmt`*, implemented in* `timingObj`*. It stores the storage and display management timing values for the current cycle. The input is the cycle timing structure, and the output is the updated cycle timing structure.*

```
    %----------------------------------------------------------------------
    % timing update
    %----------------------------------------------------------------------
    timingObj=timingObj.timing_update(cycleObj.cycle_timing);
```

This calls the method `timing_update`*, implemented in* `timingObj`*. It transfers the current cycle timing to the timing object, for later storage. The input is the cycle timing structure, and the output is the updated timing object.*

```
end  % run_cycle
```

This is the end of the cycle.

4.3.9 FAR Engine

For a particular example, the FAR engine first defines specific implementations of each of the objects, then runs the perception-action cycle, and completes any storage or display management functions.

4.3.10 Utility Functions

The utility functions object contains common utility functions. It consists of:

- Methods: hamming, compute_csigma_ellipse, TimeNowInSeconds.

4.4 The CREW Sensor

The CREW is the world's first radar test bed built specifically to allow testing of fully adaptive and cognitive algorithms. The CREW was built using an approximately $600K grant from the 2013 DURIP, sponsored by AFOSR. The CREW was designed and specified by CSL director Dr. Graeme Smith and developed by Keysight Technologies (formerly Agilent) and Millitech. The system is a four-channel multistatic radar operating in W-band. There are four pairs of transmit and receive heads allowing full distributed MIMO operation. A block diagram is provided in Figure 4.

The CREW has a fully digital back end, shown to the left in Figure 4, comprising a control PC, four analog-to-digital converters (ADCs) and four arbitrary waveform generators (AWGs). The ADCs and AWGs are connected to the PCI via a PXIe extension system, meaning that the PC is genuinely "in-the-loop" since the transfer rates are high enough that the digitized signals on all four channels can be evaluated and modifications made to the transmit waveforms in real-time. The system can be programmed using MATLAB allowing for rapid prototyping of fully adaptive and cognitive algorithms.

For the radio frequency (RF) front end, shown to the right in Figure 4, the instantaneous bandwidth of the system is 1 GHz, the transmit center frequency is 94 GHz (W-band), the effective radiated isotropic power is 50 dBW, the receiver gain is 55 dB and the receiver noise figure is ≈5 dB. The system is fully coherent across all four channels with a phase stability/error better than 1° root mean square (RMS). A two stage up/downconversion scheme is used and variation in the second local oscillator frequency allows stepped frequency processing across a 4 GHz operational bandwidth.

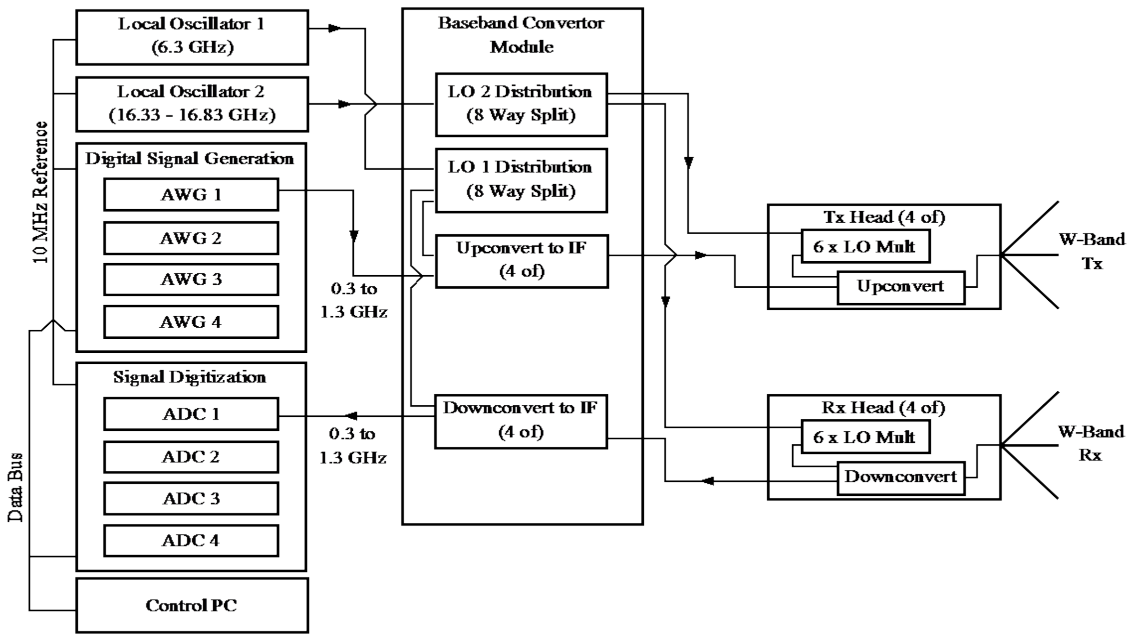


Figure 4: CREW Schematic Showing the Digital Backend, IF Stage, and W-band RF Stage

The CREW is a unique test bed able to facilitate research into fully adaptive and cognitive algorithms and distributed/multistatic operation all within the laboratory. The extremely high bandwidths result in a range resolution of a few centimeters making it possible to observe multiple scattering centers in smaller targets and easy to range-gate out the walls of the laboratory. The W-band center frequency means the narrowband assumption can be made even if the full 4 GHz bandwidth is used. As such it is easy to set-up sophisticated experiments using the CREW within CSL.

However, developing and debugging FAR algorithms while running experiments can be cumbersome and time-consuming. While developing algorithms under the FAR Phase II SBIR, it became apparent that a simulation of the CREW was needed for algorithm development and testing under controlled and reproducible conditions, with the ability to switch between simulated and experimental data sources easily. In this project, we have developed this capability.

The implementation of the CREW Sensor object consists of:

- Properties: c, transmitter (structure), receiver (structure), waveform (structure), Simulated, ConfigMaster, ConfigSlave, driver
- Methods: Sensor_CREW (constructor), get_measurement, constructWaveform, sendWaveform, receiveDataSim, receiveDataCREW

The property `c` is the speed of light and `transmitter, receiver`, and `waveform` are structures that contain parameters that characterize the CREW sensor. The parameters we can adapt are in `waveform`. For the five-parameter example, the parameters are PRF, number of pulses, pulse length, bandwidth, and transmitted power. The flag `Simulated` is set true for simulation and false for experiment. The parameters `ConfigMaster, ConfigSlave`, and `driver` are used to interface to the CREW.

The object `Sensor_CREW` implements the `get_measurement` method as shown in Figure 5. It contains a switch to toggle between simulated and CREW data. The CREW API consists of the methods `sendWaveform` and `receiveDataCrew`. These interface directly with the CREW to transmit pulses and receive echo returns. The simulation API and sensor consists of the methods `constructWaveform` and `receiveDataSim`. The method `receiveDataSim` gets target, clutter and noise parameters from `Scene_CREW`, and generates random samples of complex clutter data.

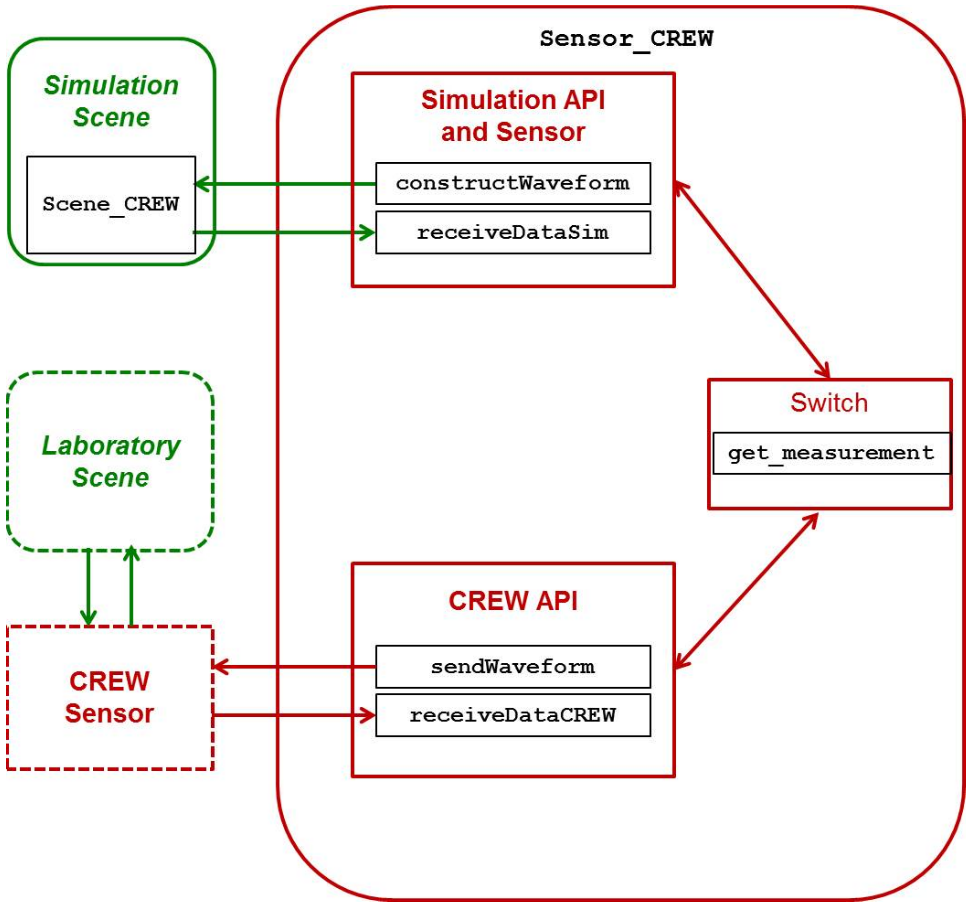


Figure 5: CREW Sensor Object

4.5 Summary

This chapter provided a description of the MATLAB-based FAR M&S architecture. The architecture is coded in MATLAB using an OOP approach. It includes a FAR engine to control the operation of the perception-action cycle and software objects that determine the next set of sensing parameters; obtain data from the sensor; process the data to track the target; and store and display the results of the sensing and tracking processes. We have developed modules that implement simulated DSRA and pre-recorded SDR data examples in [30]-[35], and the real-time CREW data examples in [36]-[39]. We have developed a simulation of the CREW and the API layers for the simulated and experimental CREW data sources to enable switching between simulated and experimental data.

5 CONCLUSIONS

The potential of cognitive approaches to enhance existing radar performance in almost all respects has led to an upsurge in research in recent years and a key gap in the Air Force’s radar M&S tools is the lack of a comprehensive, dynamic distributed radar scenario generation capability for distributed FAR systems.

In this project we have developed a MATLAB-based M&S architecture for distributed FAR radar that will enable algorithm development and testing on simulated, previously collected, and real-time streaming data. The architecture is coded in MATLAB using an OOP approach and implements the FAR framework developed in [30]-[35]. It includes a FAR engine to control the operation of the perception-action cycle and software objects that determine the next set of sensing parameters; obtain data from the sensor; process the data to track the target; and store and display the results of the sensing and tracking processes. We have developed modules that implement simulated DSRA example in [33], the pre-recorded SDR data example in [35], and the real-time CREW data examples in [36]-[40]. We have developed a simulation of the CREW and the API layers for the simulated and experimental CREW data sources to enable switching between simulated and experimental data. A demonstration was given in March 2016 for members of the NATO SET-227 Panel on Cognitive Radar.

The FAR M&S architecture developed in Phase I allows for transparent switching between the simulated and experimental CREW data sources, as well as between FAR algorithms that drive the sensing. The ability to easily interchange sensing and processing objects will allow for rapid development and testing of cognitive radar algorithms by structuring the M&S functions to avoid duplicating effort and “single point” solutions. It will enable collaboration between researchers in industry, academia, and the Air Force, as algorithms developed by different researchers can be tested and compared using consistent simulations, collected data, and laboratory conditions. In Phase II, we plan to make the FAR M&S architecture code available to members of the NATO SET-227 Panel on Cognitive Radar. Collaborations with members of this panel to develop and test algorithms on the CREW are already underway. Furthermore, several members have already begun development of their own cognitive radar test beds and our FAR M&S architecture will enable further collaboration within the panel using these data sources.

In Phase II, we also plan extend the baseline architecture to model a dynamic, distributed airborne MIMO radar FAR system using the full MIMO-CMS tool as the simulation base. This will provide a comprehensive radar scenario generation capability that will fill a key gap in the Air Force’s previously developed radar M&S tools.

6 REFERENCES

[1] Li, J. and Stoica, P., Eds., **MIMO Radar Signal Processing**, Hoboken, NJ: Wiley, 2008.

[2] San Antonio, G., Fuhrmann, D. R., and Robey, F. C., "MIMO radar ambiguity functions," *IEEE J. Selected Topics in Signal Processing*; special issue on Adaptive Waveform Design, vol. 1, no. 1, pp. 167-177, June 2007.

[3] Fishler, E., Haimovich, A., Blum, R. S., Cimini, Jr., L. J., Chizhik, D., and Valenzuela, R. A., "Spatial diversity in radars - models and detection performance," *IEEE Trans. Signal Processing*, vol. 54, no. 3, pp. 823-838, Mar. 2006.

[4] Haimovich, A., Blum, R. S., and Cimini, Jr., L. J, "MIMO radar with widely separated antennas," *IEEE Signal Processing Magazine*, vol. 25, no. 1, pp. 116-129, Jan. 2008.

[5] He, Q., Lehmann, N. H., Blum, R. S., and Haimovich, A. M., "MIMO radar moving target detection in homogeneous clutter," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 46, no. 3, pp. 1290-1300, July 2010.

[6] Wang, P., Li, H., and Himed, B., "Moving target detection using distributed MIMO radar in clutter with nonhomogeneous power," *IEEE Trans. Signal Processing*, vol. 59, no. 10, pp. 4809-4820, Oct. 2011.

[7] Chen, C.-Y. and Vaidyanathan, P. P., "MIMO radar space-time adaptive processing using prolate spheroidal wave functions," *IEEE Trans. Signal Processing*, vol. 56, no. 2, pp. 623-634, Feb. 2008.

[8] Forsythe, K. W. and Bliss, D. W., "MIMO radar waveform constraints for GMTI," *IEEE J. Selected Topics in Signal Processing*, special issue on MIMO Radar, vol. 4, no. 1, pp. 21-32, February 2010.

[9] Kantor, J. M. and Bliss, D. W., "Clutter covariance matrices for GMTI MIMO radar," in *Proc. 44th Asilomar Conf. on Signals, Systems, and Computers*, Pacific Grove, CA, pp. 1821-1826, November 2010.

[10] Fuhrmann, D. R., Browning, J. P., and Rangaswamy, M., "Signaling strategies for the hybrid MIMO phased array radar," *IEEE J. Selected Topics in Signal Processing*, special issue on MIMO Radar, vol. 4, no. 1, pp. 66-78, February 2010.

[11] Pugh, M. L. and Zulch, P. A., "RLSTAP Algorithm Development Tool for analysis of advanced signal processing techniques," in *Proc. 29th Asilomar Conf. on Signals, Systems, and Computers*, Pacific Grove, CA, pp. 1178-1182, November 1995.

[12] Ward, J., "Space-time adaptive processing for airborne radar," MIT Lincoln Laboratory Tech. Report 1015, DTIC No. ESC-TR-94-109, December 13, 1994.

[13] Guerci, J. R., **Space-Time Adaptive Processing for Radar**, Norwood, MA: Artech House, 2003.

[14] Klemm, R., **Principles of Space-Time Adaptive Processing, 3rd ed.**, London, UK: The Institution of Engineering and Technology, 2006.

[15] Zhang, Y. and Himed, B., "Bistatic space-time adaptive processing (STAP) for airborne/spaceborne applications," AFRL Tech. Report AFRL-SN-RS-TR-1999-97, May 1999.

[16] Zhang, Y. and Hajjari, S., "Bistatic space-time adaptive processing for airborne/spaceborne applications – clutter characteristics and signal processing algorithms," AFRL Tech. Report AFRL-SN-RS-TR-2001-201, Part I, Aug. 2002.

[17] Zhang, Y. and Hajjari, S., “Bistatic space-time adaptive processing for airborne/spaceborne applications – signal modeling and simulation tool for multichannel bistatic systems (SMS-MBS),” AFRL Tech. Report AFRL-SN-RS-TR-2001-201, Part II, Aug. 2002.

[18] Bell, K. L., Johnson, J. T., and Smith, G. E., “MIMO radar clutter modeling and simulation,” AFOSR Phase II Final Tech. Report, April 2016.

[19] Bell, K. L., Johnson, J. T., Baker, C. J., and Smith, G. E., “MIMO radar clutter modeling,” AFRL Tech. Report AFRL-RY-WP-TR-2012-0308, October 2012.

[20] Goodman, N. A. and Bruyere, D., “Optimum and decentralized detection for multistatic airborne radar,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 43, no. 2, pp. 806-813, April 2007.

[21] Bell, K. L., Johnson, J. T., Baker, C. J., Smith, G. E. and Rangaswamy, M., “Modeling and simulation for multistatic coherent MIMO radar,” in *Proc. 2013 IEEE Radar Conf.*, Ottawa, Canada, April 2013.

[22] Johnson, J. T., Baker, C. J., Smith, G. E., Bell, K. L., and Rangaswamy, M., “The monostatic-bistatic equivalence theorem and bistatic radar clutter,” in *Proc. European Radar Conference 2014*, Rome, Italy, October 2014.

[23] Bell, K. L., Johnson, J. T., Baker, C. J., Smith, G. E. and Rangaswamy, M., “Bistatic coherent MIMO clutter rank analysis,” in *Proc. 2015 European Signal Processing Conf.*, Nice, France, Aug. 2015.

[24] Fuster, J. M., **Cortex & Mind: Unifying Cognition**, Oxford, UK: Oxford University Press, 2003.

[25] Fuster, J. M, “The cognit: A network model of cortical representation,” *Intl. Journal of Psychophysiology*, vol. 60, no. 2, pp. 125-132, May 2006.

[26] Haykin, S., “Cognitive radar: a way of the future,” *IEEE Signal Processing Magazine,* vol. 23, no. 1, pp. 30–40, Jan. 2006.

[27] Haykin, S., Xue, Y., and Setoodeh, M. P., “Cognitive radar: step toward bridging the gap between neuroscience and engineering,” *Proc. IEEE*, vol. 100, no. 11, pp. 3102-3130, November 2012.

[28] Haykin, S., **Cognitive Dynamic Systems (Perception-Action Cycle, Radar, and Radio)**, Cambridge University Press, 2012.

[29] “Cognitive Dynamic Systems,” S. Haykin, Ed., special issue of *Proc. IEEE*, vol. 102, no. 4, Apr. 2014.

[30] Bell, K. L., Baker, C. J., Smith, G. E., and Johnson, J. T., “Fully Adaptive Radar,” AFRL Tech. Report AFRL-RY-WP-TR-2014-0072, March 2014.

[31] Bell, K. L., Baker, C. J., Smith, G. E., Johnson, J. T., and Rangaswamy, M., “Fully adaptive radar for target tracking part I: single target tracking,” in *Proc. 2014 IEEE Radar Conf.*, Cincinnati, OH, pp. 303-308, May 2014.

[32] Bell, K. L., Baker, C. J., Smith, G. E., Johnson, J. T., and Rangaswamy, M., “Fully adaptive radar for target tracking part II: target detection and track initiation,” in *Proc. 2014 IEEE Radar Conf.*, Cincinnati, OH, pp. 309-314, May 2014.

[33] Bell, K. L., Baker, C. J., Smith, G. E., Johnson, J. T., and Rangaswamy, M., “Cognitive radar for target tracking,” *IEEE J. Selected Topics in Signal Processing*; special issue on Advanced Signal Processing Techniques for Radar Applications, vol. 9, no. 8, pp.1427-1439, December 2015.

[34] Bell, K. L., Baker, C. J., Smith, G. E., Johnson, J. T., and Rangaswamy, M., “Cognitive sensor/processor system framework for target tracking,” chapter in **Biologically Inspired Radar and Sonar: Lessons from Nature**, A. Balleri, H. Griffiths, and C. Baker, eds., in press.

[35] Bell, K. L., Johnson, J. T., Smith, G. E., Baker, C. J., and Rangaswamy, M., “Cognitive radar for target tracking using a software defined radar system,” in *Proc. 2015 IEEE Radar Conf.*, Arlington, VA, pp. 1394-1399, May 2015.

[36] Smith, G. E., Cammenga, Z., Mitchell, A., Bell, K. L., Rangaswamy, M., Johnson, J. T., and Baker, C. J., “Experiments with cognitive radar,” in *Proc. IEEE Intl. Wkshp. on Comp. Adv. in Multi-Sensor Adaptive Processing (CAMSAP 2015),* Dec. 2015.

[37] Smith, G. E., Cammenga, Z., Mitchell, A., Bell, K. L., Johnson, J. T., Rangaswamy, M., and Baker, C. J., “Experiments with cognitive radar,” *IEEE Aerospace and Electronic Systems Magazine*, special issue on Waveform Diversity: Part II, vol. 31, no. 12, pp. 34-46, Dec. 2016.

[38] Mitchell, A. E., Bell, K. L., Smith, G. E. and Rangaswamy, M., “Cognitive radar adaptation through coordinate descent optimization,” in *Proc. 2016 CIE Intl. Radar Conf.*, China, Oct. 2016.

[39] Butterfield, A., Mitchell, A. E., Smith, G. E., Bell, K. L., and Rangaswamy, M., “Metrics for quantifying cognitive radar performance, in *Proc. 2016 CIE Intl. Radar Conf.*, China, Oct. 2016.

[40] Mitchell, A. E., Smith, G. E., Bell, K. L., and Rangaswamy, M., “Single target tracking with distributed cognitive radar,” to appear in *Proc. 2017 IEEE Radar Conf.,* Seattle, WA, May 2017.

[41] Gini, F. and Rangaswamy, M., Eds., **Knowledge Based Radar Detection, Tracking, and Classification**, Hoboken, NJ: Wiley, 2008.

[42] Guerci, J. R., **Cognitive Radar: The Knowledge Aided Fully Adaptive Approach**, Reading, MA: Artech House, 2010.

[43] Guerci, J., Guerci, R., Rangaswamy, M., Bergin, J., and Wicks, M., “Cognitive Fully Adaptive Radar (CoFAR),” in *Proc. 2014 IEEE Radar Conf.*, Cincinnati, OH, May 2014.

[44] Miranda, S. L. C., Baker, C. J., Woodbridge, K. D., and Griffiths, H. D., “Knowledge based resource management for multifunction radar,” *IEEE Signal Processing Magazine*, vol. 23, no. 1, pp. 66-76, 2006.

[45] Miranda, S. L. C., Baker, C. J., Woodbridge, K. D., and Griffiths, H. D., “Fuzzy logic approach for prioritization of radar tasks and sectors of surveillance in multifunction radar,” *IET Proc. Radar, Sonar, Navig.*, vol. 1, no. 2, pp. 131-141, 2007.

[46] Vespe, M., Baker, C. J., and Griffiths, H. D., “Automatic target recognition using multi-diversity radar,” *IET Proc. Radar, Sonar, Navig.*, vol. 1, no. 6, pp. 470-478, Dec 2007.

[47] Miranda, S., Baker, C. J., Woodbridge, K., and Griffiths, H. D., “Intelligent radar resource management,” chapter in **Knowledge Based Radar Detection, Tracking and Classification**, F. Gini and M. Rangaswamy, Eds., ISBN: 978-0-470-14930-0, Wiley, 2008.

[48] Aubry, A., De Maio, A., Piezzo, M., Farina, A., and Wicks, M. “Cognitive design of the receive filter and transmitted phase code in reverberating environment,” *IET Radar, Sonar and Navigation*, vol. 6, no. 9, pp. 822-833, 2012.

[49] Stinco, P., Greco, M., and Gini, F., “Spectrum sensing and sharing for cognitive radars,” *IET Radar, Sonar and Navigation*, vol. 10, no. 3, pp. 595–602, 2016.

[50] Stinco, P., Greco, M., Gini, F., and Himed B., “Cognitive radars in spectrally dense environments,” *IEEE Aerosp. Electron. Syst. Magazine*, vol. 31, no. 10, pp. 20-27, October 2016.

[51] Kershaw, D. J. and Evans, R. J., “Optimal waveform selection for tracking systems,” *IEEE Trans. Inform. Theory,* vol. 40, no. 5, pp. 1536-1550, Sep. 1994.

[52] Fuhrmann, D., “Active-testing surveillance systems, or, playing twenty questions with radar," in *Proc. 11th Annual Adaptive Sensor and Array Processing (ASAP) Workshop*, MIT Lincoln Laboratory, Lexington, MA, Mar. 2003.

[53] Sira, S. P., Papandreou-Suppappola, A., and Morrell, D., “Dynamic configuration of time-varying waveforms for agile sensing and tracking in clutter,” *IEEE Trans. Signal Process*ing, vol. 55, no. 7, pp. 3207-3217, Jul. 2007.

[54] Hurtado, M., Zhao, T., and Nehorai, A., “Adaptive polarized waveform design for target tracking based on sequential Bayesian inference,” *IEEE Trans. Signal Processing*, vol. 56, no. 13, pp. 1120-1133, Mar. 2008.

[55] Sira, S. P., Li, Y., Papandreou-Suppappola, A., Morrell, D., Cochran, D., and Rangaswamy, M., “Waveform-agile sensing for tracking,” *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 53-64, Jan. 2009.

[56] Hernandez, M. L., Kirubarajan, T., and Bar-Shalom, Y., “Multisensor resource deployment using posterior Cramér-Rao Bounds,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 40, no. 2, pp. 399-416, Apr. 2004.

[57] Kreucher, C., Hero, A. O., Kastella, K., and Chang, D., “Efficient methods of non-myopic sensor management for multitarget tracking,” in *Proc. 43rd IEEE Conf. Decision and Control*, Atlantis, Bahamas, pp. 722-727, December 2004.

[58] Kreucher, C., Kastella, K. , and Hero, A. O., “Sensor management using an active sensing approach,” *Signal Processing*, vol. 85, no. 3, pp. 607-624, March 2005.

[59] Kreucher, C. M., Hero, A. O., Kastella, K. D., and Shapo, B., “Information-based sensor management for simultaneous multitarget tracking and identification,” in *Proc. 13th Ann. Conf. on Adaptive Sensor Array Processing*, MIT Lincoln Laboratory, Lexington, MA, June 2005.

[60] Kreucher, C., Hero, A. O., and Kastella, K., “A comparison of task driven and information driven sensor management for target tracking,” in *Proc. 44th IEEE Conf. Decision and Control*, Seville, Spain, pp. 4004-4009, December 2005.

[61] Kreucher, C. M., Hero, A. O., Kastella, K. D., and Morelande, M. R., “An information-based approach to sensor management in large dynamic networks”, *Proc. IEEE,* vol. 95, no. 5, pp. 978-999, May 2007.

[62] Tharmarasa, R., Kirubarajan, T., and Hernandez, M. L., “Large-scale optimal sensor array management for multitarget tracking,” *IEEE Trans. Syst., Man, Cybern. C: Appl. Rev.*, vol. 37, no. 5, pp. 803-814, Sep. 2007.

[63] Chong, E. K. P., Kreucher, C. M., and Hero, A. O. “Monte-Carlo-based partially observable Markov decision process approximations for adaptive sensing,” in *Proc. 9th Intl. Wkshp. Discrete Event Systems*, Goteborg, Sweden, pp. 173-180, May 2008.

[64] Chavali, P. and Nehorai, A., “Scheduling and power allocation in a cognitive radar network for multiple-target tracking,” *IEEE Trans. Signal Process.*, vol. 60, no. 2, pp. 715-729, Feb. 2012.

[65] Romero, R. A. and Goodman, N. A., “Cognitive radar network: cooperative adaptive beamsteering for integrated search-and-track application,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 49, no. 2, pp. 915-931, Apr. 2013.

[66] Charlish, A. and Hoffmann, F., “Anticipation in cognitive radar using stochastic control,” in *Proc. 2015 IEEE Radar Conf.*, Arlington, VA, pp. 751-756, May 2015.

[67] Saksena, A. and Wang, I-J., “Dynamic ping optimization for surveillance in multistatic sonar buoy networks with energy constraints,” in *Proc. 47th IEEE Conf. Decision and Control*, Cancun, Mexico, pp. 1109-1114, Dec. 2008.

[68] Wakayama, C. Y. and Grimmett, D. J., “Adaptive ping control for track-holding in multistatic active sonar networks,” in *Proc. 13th Intl. Conf. on Information Fusion*, Edinburgh, UK, July 2010.

[69] Wakayama, C. Y., Grimmett, D. J., and Zabinsky, Z. B., “Forecasting probability of target presence for ping control in multistatic sonar networks using detection and tracking models,” in *Proc. 14th Intl. Conf. on Information Fusion*, Chicago, IL, July 2011.

[70] Holdereid, M. W., Baker, C. J., Vespe, M., and Jones, G., “Understanding signal design during the pursuit of aerial insects by echo locating bats: tools and applications,” *Integrative and Comparative Biology*, vol. 48, pp. 78-84, May 2008.

[71] Baker, C. J. and Griffiths, H. D., “Biologically inspired waveform diversity,” chapter in **Waveform Design and Diversity for Advanced Radar Systems**, F. Gini, A. De Maio, and L. Patton, Eds., IET publishing, Aug. 2012.

[72] Balleri, A., Griffiths, H. D., Baker, C. J., Woodbridge, K., and Holderied, M. W., “Analysis of acoustic echoes from a bat-pollinated plant species: insight into strategies for radar and sonar target classification,” *IET Proc. Radar, Sonar, Navig.*, vol. 6, no. 6, pp. 536-544, July 2012.

[73] Baker, C. J. and Smith, G. E., “Aspects of cognition and echolocation,” in *Proc. 2012 IEEE Antennas and Propagation Society International Symposium (APSURSI)*, pp. 1-2, 8-14 July 2012.

[74] Baker, C. J., Smith, G. E., Balleri, A., Holderied, M. and Griffiths, H. D., “Biomimetic echolocation with application to radar and sonar sensing,” *Proc. IEEE*, vol. 102, no. 4, pp. 447–458, Apr. 2014.

[75] Baker, C. J., Smith, G. E., Balleri, A., Holderied, M. and Griffiths, H. D., “Sensing, cognition, and engineering application [further thoughts],” *Proc. IEEE*, vol. 102, no. 4, p. 459, Apr. 2014.

[76] Smith, G. E. and Baker, C. J., “Echoic flow for radar and sonar,” *IET Electronics Letters*, vol. 48, no. 18, pp. 1160-1161, August 2012.

[77] Smith, G. E. and Baker, C. J., “Echoic flow for autonomous navigation,” in *Proc. of Radar 2012, The International Conference On Radar*, 2012.

[78] Frankford, M., Majurec, N., and Johnson, J. T., “Software-defined radar for MIMO and adaptive waveform applications”, *2010 IEEE Radar Conf.*, Washington, DC, pp. 724-728, May 2010.

[79] Frankford, M., Johnson, J. T., and Ertin, E., “Including spatial correlations in the statistical MIMO radar target model,” *IEEE Signal Processing Letters*, vol. 17, no. 6, pp. 575-578, June 2010.

[80] Frankford, M., Stewart, K. B., Majurec, N., and Johnson, J. T., “Numerical and experimental studies of target detection with MIMO radar,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 50, no. 2, pp. 1569-1577, April 2014.

[81] Stewart, K., Frankford, M., Johnson, J. T., and Ertin, E., “MIMO radar target measurements,” in *Proc. 45th Asilomar Conf. Signals, Systems, and Computers*, Pacific Grove, CA, pp. 1067-1071, Nov. 2011.

[82] Park, J., “Multi-frequency radar signatures of human motion: measurements and models,” Ph.D. Dissertation, The Ohio State University, 2012.

[83] Park, J., Johnson, J. T., Majurec, N., Frankford, M., Culpepper, E., Reynolds, J., Tenbarge, J., and Westbrook, L., “Software defined radar studies of human motion signatures,” in *Proc. 2012 IEEE Radar Conf.*, Atlanta, GA, pp. 596-601, May 2012.

[84] Park, J., Johnson, J. T., Majurec, N., Frankford, M., Stewart, K., Smith, G., and Westbrook, L., “Simulation and analysis of polarimetric radar signatures of human gaits,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 50, no. 3, pp. 2164-2175, July 2014.

[85] Oechslin, R., Smith, G. E. , Aulenbacher, U., Rech, K., Hinrichsen, S., Bell, K. L. and Wellig, P., “Cognitive radar testbed development,” in *Proc. 50th Asilomar Conf. on Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 2016.

[86] Christiansen, J. M., Smith, G. E., and Olsen, K. E., “USRP based cognitive radar testbed," to appear in *Proc. 2017 IEEE Radar Conf.*, Seattle, WA, May 2017.

[87] Stone, L. D., Streit, R. L., Corwin, T. L., and Bell, K. L., **Bayesian Multiple Target Tracking, 2nd Ed.**. Norwood, MA: Artech House, 2014.

[88] Ristic, B., Arulampalam, S., and Gordon, N., **Beyond the Kalman Filter: Particle Filters for Tracking Applications**, Boston, MA: Artech House, 2004.

[89] Bar-Shalom, Y., Li, X. R., and Kirubarajan, T., **Estimation with Applications to Tracking and Navigation**, New York, NY: Wiley, 2001.

[90] Van Trees, H. L., Bell, K. L., and Tian, Z., **Detection, Estimation, and Modulation Theory, Part I, 2nd Ed.,** Hoboken, NJ: Wiley, 2013.

[91] Van Trees, H. L. and Bell, K. L., Eds., **Bayesian Bounds for Parameter Estimation and Nonlinear Filtering/Tracking**, Piscataway, NJ: Wiley-IEEE Press, 2007.

[92] Zarnich, R. E., Bell, K. L., and Van Trees, H. L., “A unified method for measurement and tracking of multiple contacts from sensor array data,” *IEEE Trans. Sig. Proc.*, vol. 49, no.12, pp. 2950 –2961, Dec. 2001.

[93] Bell, K. L., “MAP-PF position tracking with a network of sensor arrays,” in Proc. *2005 IEEE Intl. Conf. on Acoust., Speech, Sig. Proc. (ICASSP `05)*, Philadelphia, PA, vol. IV, pp. 849-852, March 2005.

[94] Bell, K. L. and Pitre, R. “MAP-PF 3D position tracking using multiple sensor arrays,” in Proc. *Fifth IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM 2008)*, Darmstadt, Germany, pp. 238-242, July 2008.

[95] Bell, K. L., Zarnich, R. E., and Wasyk, R., “MAP-PF wideband multitarget and colored noise tracking,” in *Proc. 2010 IEEE Intl. Conf. on Acoust., Speech, Sig. Proc. (ICASSP `10)*, Dallas, TX, pp. 2710-2713, March 2010.

[96] Bell, K. L., “MAP-PF multi-mode tracking for over-the-horizon radar,” in *Proc. 2012 IEEE Radar Conf.*, Atlanta, GA, pp. 326-331, May 2012.

[97] Bell, K. L. and Zarnich, R. E., “MAP-PF multitarget tracking with propagation modeling uncertainties,” in *Proc. 47th Asilomar Conf. Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 2013.

[98] Zangwill, W. I., **Nonlinear Programming: A Unified Approach**, Englewood Cliffs, NJ: Prentice-Hall, 1969.

LIST OF ACRONYMS, ABBREVIATIONS, AND SYMBOLS

ACRONYM	DESCRIPTION
A2/AD	anti-access/area denial
ADC	analog-to-digital converter
AFOSR	Air Force Office of Scientific Research
AFRL	Air Force Research Laboratory
API	application programming interface
AWG	arbitrary waveform generator
BCRLB	Bayesian Cramér-Rao lower bound
BIM	Bayesian information matrix
BLR	Bayesian likelihood ratio
BLRT	Bayesian likelihood ratio test
CPI	coherent processing interval
CR	Cognitive radar
CREW	Cognitive Radar Engineering Workspace
CSL	Cognitive Sensing Laboratory
DSRA	distributed sensor resource allocation
DURIP	Defense University Research Instrumentation Program
EFIM	expected Fisher information matrix
ESL	ElectroScience Laboratory
FAR	fully adaptive radar
FFR	feed-forward radar
FIM	Fisher information matrix
GMTI	ground moving target indicator
IF	intermediate frequency
ILR	integrated likelihood ratio
LRDT	likelihood ratio detection and tracking
M&S	modeling and simulation
MAP	maximum a posteriori
MAP-PF	maximum a posteriori penalty function
MBET	monostatic-bistatic equivalence theorem
MIMO	multiple input multiple output
MIMO-CMS	MIMO radar clutter modeling and simulation
ML	maximum likelihood
MMSE	minimum mean square error
MSE	mean square error

NATO	North Atlantic Treaty Organization
OOP	object oriented programming
OSU	The Ohio State University
PA	perception-action
PDF	probability density function
PC-BIM	predicted conditional Bayesian information matrix
PC-CRLB	predicted conditional Cramér-Rao lower bound
PIM	predicted information matrix
PRF	pulse repetition frequency
RCS	radar cross section
RF	radio frequency
RLSTAP	Research Laboratory Space Time Adaptive Processing
RMS	root mean square
SDR	software defined radar
SET	Sensors Electronics Technology
SIMO	single input multiple output
SISO	single input single output
SMS-MBS	Signal Modeling and Simulation Tool for Multichannel Bistatic Systems
SNR	signal-to-noise ratio
SPC	Signal Processing Consultants, Inc.
STAP	space-time adaptive processing